Sharon Boren • Larry Hovey • Kathleen Hovey

An ATARI®
In The Classroom
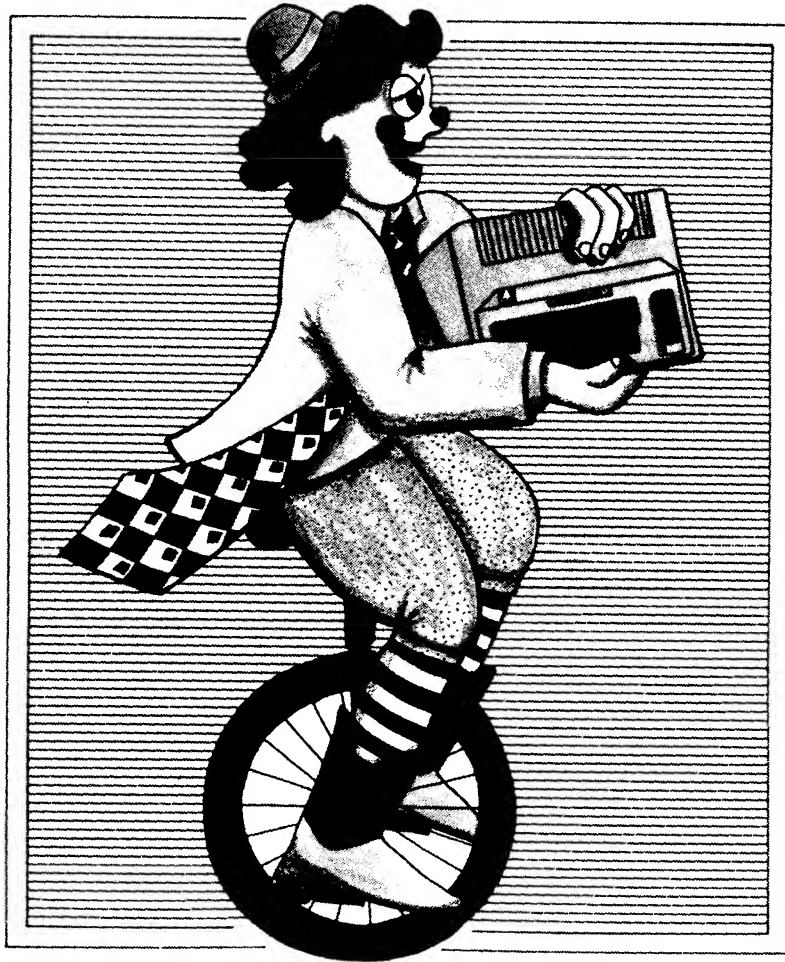
dilithium Press

Teacher's Guide

dP dilithium Press

An ATARI® in the

Classroom

An ATARI® in the Classroom: Teacher's Guide

Sharon Boren

Larry Hovey

Kathleen Hovey

10   9   8   7   6   5   4   3   2   1

Some Thoughts About Computers in the Classroom

As individuals, we have seen our world greatly changed by the advent of computers in our lives. The businesses we deal with, the travel schedules we follow, the stoplights we drive through, and most aspects of our lives are influenced by computers. Although we are tremendously affected by computers, we too often view the machines as mysterious and incomprehensible, and the people who work with computers as mathematical wizards.

A decade ago, few of us could have imagined that computers would one day become an integral part of our classrooms. However, the age of educational computers is now upon us. Computers will be used in schools in a variety of ways. In particular, they will help us to MANAGE clerical and administrative details, such as evaluating, grading, attendance accounting, making payrolls, and taking inventories. Computers will also ASSIST INSTRUCTION through programs that will teach or reinforce concepts and skills. Perhaps most interesting, students will learn to control computers by DESIGNING PROGRAMS which tell the machines what to do. *An ATARI for Kids* focuses on this component of computer education—HELPING KIDS TO BECOME COMPUTER PROGRAMMERS—at home and at school.

One of the main reasons that computers are now practical in the classroom is that they have evolved from extremely expensive, cumbersome systems, to relatively inexpensive, small MICROCOMPUTERS. It is the microcomputer that is rapidly finding its way into many schools and homes, and forcing a redesign of our school curriculum—a curriculum that requires teachers to take a leadership role in helping students understand and use computers in their lives.

Some teachers have already begun to explore and use computers, while others are left with many concerns and reservations. This book, *An ATARI for Kids*, is intended to help teachers—even those unfamiliar with computers—teach their third through eighth grade students to design computer programs. Furthermore, we assume that what kids learn from instruction in these pages and in your classroom will be equally applicable at home.

As you introduce your students to the exciting world of the microcomputer, may you and they encounter successful and rewarding experiences!

THE BEST OF LUCK TO YOU AND YOUR STUDENTS WITH YOUR COMPUTER ENDEAVORS!

Sharon, Kathi, and Larry

TABLE OF CONTENTS

Using the Teacher's Guide

There is tremendous variety in the state of computer instruction in our schools. Computer education differs widely with the availability of computers and software (programs that run computers), as well as experience levels of teachers and students. For this reason *An ATARI for Kids* does not attempt to prescribe a formal classroom program. Rather, it seeks to provide some carefully thought out, flexible activities and ideas that can fit a variety of classroom situations.

Objectives. The main objective of *An ATARI for Kids* is to assist students in becoming capable computer programmers. These materials also encourage students to explore and be creative with the computer—through problem-solving, designing their own keyboard games, and writing programs that utilize graphics and sound.

The secondary objective of *An ATARI for Kids* is to help those of you who are new to computers to become familiar with simple programming techniques, and thereby assist your students as they work with the ATARI computer.

In order to successfully understand computer programming and to guide your students as they program, we suggest you follow these steps: read the appropriate student text pages with the accompanying information here; use the ATARI to work the student activity sheets; and relax and enjoy the fact that you can better understand and control the computers in your world.

Grade Level Placement. The students text and activity worksheets begin at approximately a third to fourth grade reading level, and math within the first four components is that which the above average third grader and average fourth grader should be able to handle. After the fourth component, the book increases in difficulty as more programming techniques are introduced. It isn't intended that younger students work through the entire text and worksheets during their first year, although it is possible. Rather, they can acquire a sound base for creative programming the first year, and continue where they left off the following year.

Format. This TEACHER'S GUIDE is part of a three-book set. *An ATARI for Kids* is the student text. *An ATARI in the Classroom: ACTIVITY WORKBOOK* contains student worksheets. This TEACHER'S GUIDE includes general information for the teacher; a chapter-by-chapter explanation of student assignments, new vocabulary, and objectives; and answers for the activity worksheets.

The activity worksheet format allows for very flexible classroom instruction. If only one computer is available, the activity sheets can be used as seatwork, with the ATARI being used individually by students to check their work. (Often, two or three students can use the computer at the same time since students tend to assist and learn from each other.) If greater numbers of computers are available, *An ATARI for Kids* lends itself to small group, or even large group instruction.

Computer Language. Many different LANGUAGES can be used to program computers. We have chosen BASIC (Beginners All-Purpose Symbolic Instruction Code) as the computer language for *An ATARI for Kids*. BASIC was selected

because it can be learned readily by third through eighth graders, and because it is very popular, thus having a wide transferability outside the classroom.

Assignments and Evaluation. Because *An ATARI for Kids* is designed to fit a variety of classroom situations, there is no one best way to make assignments. The activity worksheets are the heart of the program, and are to be assigned to individuals, small groups, or the entire class in a way that best fits your classroom needs. A STUDENT PROGRESS CHART (found at the end of this Teacher's Guide) can be duplicated and used to record the completion of student assignments.

Formal evaluation is not a part of *An ATARI for Kids*. This is best designed by the individual teacher to fit the particular situation. Most teachers will find that students are highly motivated when working with computers, and that students do a great deal of self-evaluation just because they want to improve their knowledge and skills. This personal evaluation and growth occurs as they check their work on the ATARI computer, compare ideas with their peers, and interact with you.

Of course, formal evaluation and grading may be a necessary part of your school situation. If so, evaluate your students' work with *An ATARI for Kids* in a way that best meets your needs.

An ATARI for Kids does have an EVALUATE YOURSELF exercise at the end of the seven student text components. These are meant to encourage student feedback concerning the worksheets and text. We hope you can use this information to improve your computer education program. The authors and dilithium Press would certainly appreciate your views, and those of your students, in improving *An ATARI for Kids*.

Sequencing. *An ATARI for Kids* is sequentially designed so that later chapters build on previous ones. However, it is possible to do some "skipping about" the text and worksheets, particularly for those individuals who have had some previous experience with programming. In this respect, the chapters dealing with graphics and sound are highly motivating, and can be a fun place to begin for those students with some skills and confidence with computers.

Equipment. *An ATARI for Kids* can be used with the ATARI 400, 800 or XL computers. Although there are several differences between the machines, all will work equally well with this program.

Other equipment in addition to the computer is necessary. A BASIC language cartridge is required in order to "speak" to some models. A BASIC program is built into the XL models. Also, either a television or a monitor is necessary. A color screen is preferred, especially to take advantage of ATARI's excellent graphic capabilities, but a black and white is also workable. An ATARI cassette tape recorder (your own recorder will not work), or a disk drive is necessary for saving and loading programs. (Cassette tapes or diskettes will also be needed.)

Other equipment, like a printer, paddles/joy stick controllers for games, and additional commercial programs (software) are useful, but are not required for *An ATARI for Kids*.

Objectives

Chapter 1   The student will understand what a microcomputer is, and be able to identify its four basic parts—keyboard, screen, cassette tape or disk drive unit, and brain (internal circuitry).

Chapter 2   The student will become familiar with the functioning of ATARI's keyboard.

Chapter 3   a. The student will learn how to turn on ATARI.
             b. The student will understand the writing and symbols that first appear on the screen when ATARI is turned on.
             c. The student will recognize BASIC as a language that ATARI can understand.

Chapter 4   a. The student will become familiar with ATARI's special keys and how they work.
             b. The student will learn that the SHIFT and CTRL keys must be held down while pressing another key to cause certain symbols, graphics, or functions to occur.

Chapter 5   The student will learn how to correct typing mistakes by using the screen editing feature.

Chapter 6   a. The student will learn what a program is and what a programmer does.
             b. The student will understand ways to load (input) a program into ATARI by using the LOAD command.
             c. The student will learn to erase ATARI's memory by typing NEW.

Chapter 7   a. The student will learn that a program consists of statements which are preceded by line numbers.
             b. The student will learn to write programs using the statements PRINT, GOTO, and END.
             c. The student will learn to use the RUN command to execute a program.

Chapter 8   a. The student will learn to save and load a program with the aid of a storage device—cassette tape recorder or disk drive system.
             b. The student will learn to use the LIST command to display any program in ATARI's memory.

Chapter 9	The student will learn BASIC symbols for performing six types of arithmetic.
Chapter 10	The student will learn how to solve arithmetic equations in DIRECT/IMMEDIATE MODE.
Chapter 11	a. The student will understand the order in which ATARI performs arithmetic. b. The student will learn how to write a PRINT statement in an abbreviated form.
Chapter 12	The student will understand that ATARI can perform other operations besides arithmetic.
Chapter 13	a. The student will understand the necessity of solving a problem through utilization of a step-by-step ALGORITHM. b. The student will understand how a FLOW CHART uses an algorithm and is a pre-step to writing successful programs.
Chapter 14	The student will understand how a single ALTERNATIVE-DECISION STEP works in a flow chart.
Chapter 15	The student will understand how a DOUBLE-ALTERNATIVE DECISION step works in a flow chart.
Chapter 16	The student will understand that a LOOP can be used to repeat certain steps in a program.
Chapter 17	The student will learn how to take the algorithm steps from a flow chart and code them in a BASIC program.
Chapter 18	The student will learn how to program the computer to print whole EQUATIONS.
Chapter 19	a. The student will learn how to use COMMAS and SEMI-COLONS to print equations on one screen line. b. The student will understand the layout of the PRINT ZONES on the screen and how a comma or semi-colon affects the output.
Chapter 20	a. The student will understand the basics of how ATARI's memory operates. b. The student will learn what a NUMERIC VARIABLE is, and how to properly name it.

c. The student will learn how to use the LET statement to assign a value to a variable address.

Chapter 21   a. The student will understand the difference between the AD-DRESS and the CONTENTS of a variable.
b. The student will learn how to use variables in a program to print both variable names and variable contents.

Chapter 22   The student will learn how to use variables in a program to print and solve arithmetic equations.

Chapter 23   The student will understand the ramifications of using a LET statement before a PRINT statement in a program, and vice versa.

Chapter 24   The student will learn how to use colons and commas to shorten a program.

Chapter 25   a. The student will understand what type of numbers ATARI can work with.
b. The student will understand how to read E NOTATIONS (FLOATING POINT NOTATIONS).

Chapter 26   The student will learn how to use FOR-NEXT loops in a program to create COUNTER-CONTROLLED LOOPING.

Chapter 27   The student will learn how to use the STEP statement in a program to make ATARI count in number patterns.

Chapter 28   The student will learn how to use a COUNTER in a program to keep track of how many times a loop has been executed.

Chapter 29   a. The student will learn how to program ATARI to clear the screen during the execution of a program.
b. The student will learn how to use a FOR-NEXT time loop to slow down the printing of output on the screen.
c. The student will learn how to use colons to place all of a FOR-NEXT statement on a single line.

Chapter 30   The student will learn how to program BLINKING OUTPUT.

Chapter 31   a. The student will learn how to use the BYE and CONT commands.
b. The student will learn how to use the STOP statement.
c. The student will learn the function of the MEMO PAD MODE.

Chapter 32	a. The student will understand the technique of debugging.
	b. The student will understand three basic types of computer errors.
Chapter 33	a. The student will understand what a STRING VARIABLE is and the correct way to use it in a program.
	b. The student will learn how to DIMENSION a string variable.
Chapter 34	a. The student will understand what INTERACTIVE PROGRAMMING is.
	b. The student will learn how to use the INPUT statement in a program to interact with the computer.
Chapter 35	a. The student will learn the BASIC signs used to make comparisons and how to use them in a program with the IF-THEN statement.
	b. The student will understand how to use the COMPLEMENT of a question in a program using the IF-THEN statement.
Chapter 36	a. The student will understand how ATARI compares letters to alphabetize them.
	b. The student will learn how to program ATARI to alphabetize words.
Chapter 37	The student will learn how to effectively document a program by using REM statements.
Chapter 38	The student will learn how to utilize the READ-DATA statements in a program.
Chapter 39	The student will learn a process of programming geared to solving problems.
Chapter 40	The student will learn how to write a CONVERSION program.
Chapter 41	The student will learn how to use the RND and INT functions in a program to accomplish certain tasks.
Chapter 42	The student will learn how to use the SOUND statement to program ATARI to make sounds and music.
Chapter 43	The student will learn how to use the GRAPHICS, COLOR, PLOT, and DRAWTO statements to produce simple graphics.

Chapter 44	a. The student will learn how to use the SETCOLOR statement to produce some advanced graphics.
	b. The student will learn to create a graphics screen without the text window.
	c. The student will learn to combine both graphics and sound in a single program.
Chapter 45	a. The student will understand the three basic types of computer games.
	b. The student will learn how to write a game program that is USER-FRIENDLY.
Chapter 46	The student will understand that programming a computer can be a stimulating, creative experience.

Vocabulary

Chapter 1	ATARI	
	BRAIN	
	CASSETTE TAPE RECORDER	
	DISK DRIVE	
	KEYBOARD	
	MICROCOMPUTER	
	SCREEN	
Chapter 2	CURSOR CONTROL KEY	
	FUNCTION KEYS	
	GRAPHICS KEYS	
	LETTER KEYS	
	NUMBER KEYS	
	SPECIAL SYMBOL KEYS	
Chapter 3	BASIC	
	COMPUTER LANGUAGE	
	CURSOR	
	READY	
Chapter 4	BREAK MESSAGE	
	HOME	
Chapter 5	CURSOR CONTROL KEYS	
	DELETE	
	ERROR MESSAGES	
	INSERT	
	SCREEN EDITING	
Chapter 6	LOAD	
	MEMORY	
	NEW	
	PROGRAM	
	PROGRAMMER	

Chapter 1 We Have an ATARI in Our Classroom

STUDENT ASSIGNMENTS
Student Text p. 4
Activity Worksheets (none)

VOCABULARY
ATARI—A microcomputer made by ATARI, a Warner Communications Company.
Brain—The central processing unit and memory bank which make up the internal circuitry of the Atari computer.
Cassette tape recorder—A device used to store information from the computer memory, or to input information to the computer memory.
Disk drive—A device used to store information from the computer memory, or to input information to the computer memory.
Keyboard—The part of the computer used to type in information (input) to the computer brain.
Microcomputer—A compact portable computer suitable for school or home use. (Looks much like a typewriter keyboard.)
Screen—A television or monitor used to display the information from the computer.

OBJECTIVE
The student will understand what a microcomputer is, and will be able to identify its four basic parts—keyboard, screen, cassette tape or disk drive, and brain (internal circuitry).

OVERVIEW AND NOTES TO THE TEACHER
Computers, which were once very large and expensive, are now relatively inexpensive and small enough to be portable. These microcomputers are rapidly becoming common in the classroom and in many homes. The main parts of the computer which the student will be concerned with are the keyboard, screen, "brain," and cassette tape recorder or disk drive.

The ATARI computers are very sturdy, and will hold up well under normal classroom usage. However, caution the students to NEVER take food or drinks around the machines. A spill inside the computer can cause many problems. It's also wise to use a dust cover over the ATARI when it is not being used.

Chapter 2 ATARI's Keyboard

STUDENT ASSIGNMENTS
Student Text pp. 5-9
Activity Worksheets (none)

VOCABULARY
Cursor control key—Holding the control key while striking an arrow key allows the user to move the cursor in the direction of the arrow.
Function keys—Keys which control the mechanical operations of the keyboard such as shift, delete/backspace, break and return.
Graphics keys—The keys which allow graphics symbols to be made.
Letter keys—The alphabet letter keys.
Number keys—The keys which control the numerals on the top line of the keyboard.
Special symbol keys—These are keys such as +, −, *, ↓, =, ;, :, etc.

OBJECTIVE
The student will become familiar with the functioning of ATARI's keyboard

OVERVIEW AND NOTES TO THE TEACHER
This chapter familiarizes students with ATARI's keyboard and introduces them to the types of keys on the keyboard.

Many classrooms will not have enough computers to give students as much practice at the keyboard as desirable. However, students can have an illustration of the keyboard to keep and use at their desks. (A keyboard illustration is located in the front of each Activity Workbook, or duplicates can be made from the master at the end of this book.) It is sometimes helpful to enlarge the keyboard into a chart to be placed above a computer. Also, games or interest centers can be developed to help students become familiar with the keyboard. Furthermore, an old typewriter can be useful to help students learn key positions.

Chapter 3 Turning On the ATARI

STUDENT ASSIGNMENTS
Student Text pp. 10-11
Activity Worksheets (none)

VOCABULARY
BASIC—(Beginners All-purpose Symbolic Instruction Code) The most popular language used with microcomputers with wide applications outside the class-room.

Computer language —sets of symbols used to communicate with the computer.

Cursor—The square of light appearing on the television screen marking the location where data will next appear.

READY—The signal printed on the television screen when ATARI is ready to receive input.

OBJECTIVES
a.   The student will learn how to turn on ATARI.

b.   The student will understand the writing and symbols that first appear on the screen when ATARI is turned on.

c.   The student will recognize BASIC as a language that ATARI can understand.

OVERVIEW AND NOTES TO THE TEACHER
The first step in using the ATARI is learning how to turn it on and off. The student is also introduced to initial forms of communication with the computer—the computer says "READY" when properly set to interact with the student, and the cursor is displayed to show the screen location where interaction will occur. Also, the student is introduced to BASIC as the language to be used to communicate with ATARI.

Chapter 4 Using ATARI's Special Keys

STUDENT ASSIGNMENTS
Student Text pp. 12-14
Activity Worksheets p. 1 (Exploring ATARI's Keyboard #1)

VOCABULARY
Break Message—The message displayed on the screen, after the BREAK key is struck, describing the line location where the program has stopped.

Home—The position in the upper left-hand corner of the screen where the cursor starts, or to which it returns under certain conditions.

OBJECTIVES
a. The student will become familiar with ATARI's special keys and how they work.

b. The student will learn that the [SHIFT] and [CTRL] keys must be held down while pressing another key to cause certain symbols, graphics of functions to occur. The control key [CTRL] is marked [CONTROL] on some ATARI models. The keys have the same function on all models. This book refers to these as the [CTRL] key.

OVERVIEW AND NOTES TO THE TEACHER
The students are introduced to a variety of Special Keys and their functions. The System Keys (the keys on the far right row), are used with commercially made programs and for other purposes, but will not be used by students learning to write programs.

For large group instruction, the keyboard drawing (at the end of this book) can be used as a master for making student handouts, or for making a transparency for the overhead projector. (Similar keyboard drawings are in the front of each Activity Workbook.)

Chapter 5 Fixing Typing Mistakes

STUDENT ASSIGNMENTS
Student Text pp. 15-17
Activity Worksheets pp. 2-9 (Exploring ATARI's Keyboard#2, #3, #4, #5, #6, #7)

VOCABULARY
Cursor control keys—The keys which allow the cursor to be moved around the screen without erasing what is written on the screen.
Delete—Tells the computer to erase a character or line.
Error messages—ATARI's way of telling you that the computer does not understand what you want it to do.
Insert—Creates space within a line to allow the addition of new or corrected material.
Screen editing—Feature of the ATARI which allows you to correct the text on the screen before the program has been entered into the memory.

OBJECTIVE
The student will learn how to correct typing mistakes by using the screen editing feature.

OVERVIEW AND NOTES TO THE TEACHER
The students will learn how to control the screen editing keys to correct their programs and will do Activity Worksheets to reinforce what they have learned so far.

Screen editing works like this: The cursor must be moved to the place of the error. We hold the CTRL key while pressing the cursor control (arrow) key for the direction we want. Doing this will allow the cursor to be moved anywhere on the screen without erasing any of the writing. Then the error can be corrected in one of the following ways:
1) Press the space bar to erase the mistake.
2) Type over the mistake.
3) Use the DELETE BACK S key. This will erase the letter on the left side of the cursor. Then type the correct letter. You can erase the entire line by pressing the SHIFT key and DELETE BACK S.
4) Use the INSERT > key. Press CTRL and INSERT > and ATARI will add a space to the right of the cursor. A new line may be added by pressing SHIFT and INSERT >.

Chapter 6 Becoming a Programmer

STUDENT ASSIGNMENTS

Student Text p. 20

Activity Worksheets pp. 10-11 (Component 1 Fun Page and Evaluate Yourself)

VOCABULARY

LOAD—The program command used to bring programs from a cassette tape or diskette into the computer's memory.

Memory—The part of the computer which stores information for future used.

NEW—The program command used to erase unwanted programs from the computer's memory.

Program—The set of directions that tells a computer what to do.

Programmer—The person who writes computer programs.

OBJECTIVES

a.   The student will learn what a program is and what a programmer does.

b.   The student will understand ways to load (input) a program into ATARI by using the LOAD command.

c.   The student will learn to erase ATARI's memory by typing NEW.

OVERVIEW AND NOTES TO THE TEACHER

This chapter completes the first component of *An ATARI in the Classroom*, and introduces the main emphasis of the book—learning how to write programs to control computers.

It is important that students get into the habit of using the NEW command before they begin a new program. Otherwise it is possible to mix portions of wanted and unwanted programs. If this does occur, bring the mixed programs to the screen by typing LIST and pressing RETURN . Eliminate the unwanted program lines by typing only the unwanted line numbers and pressing RETURN after each number.

Sometimes the combined programs are too long to be viewed on the screen at one time. If so, you can "freeze" the screen by pressing CTRL and ⁞ at the same time. Unfreeze the screen by pressing the same keys.

Chapter 7 Teaching ATARI Simple Tricks

STUDENT ASSIGNMENTS
Student Text pp. 22-26
Activity Worksheets pp. 12-17 (Programming Your ATARI #1, #2, #3, #4, #5)

VOCABULARY
END—The program statement that tells ATARI a program run is completed. If END is not placed on the last line of a program, ATARI will assume it is there.

GOTO—The program statement that directs the computer to jump to a specified line in the program.

Input—Data or information that goes into the computer from the keyboard, cassette recorder or disk drive.

Line Number—Any number from 1 to 32767 which precedes a program statement.

Output—Data or information that comes out from the computer to the screen, printer, cassette recorder, or disk drive.

PRINT—The program statement that tells ATARI to print something on the screen. The resulting output may be letters, numbers, equations, the results of arithmetic calculations, etc. (A question mark, ?, may be used as an abbreviation for the PRINT statement.)

RUN—The command that tells the computer to execute or "do" the program.

Statement—An expression in the BASIC language that tells the computer to do something (GOTO, PRINT, FOR-NEXT). ("Command" and "statement" are often used interchangeably.)

OBJECTIVES
a.   The student will learn that a program consists of statements which are preceded by line numbers.
b.   The student will learn to write programs using the statements PRINT, GOTO, and END.
c.   The student will learn to use the RUN command to execute a program.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students begin to write computer programs. They learn several program statements and commands, and how to form statements into a program by using line numbers.

Program commands (e.g., RUN and SAVE) are not preceded by a line number. Program statements (e.g., GOTO, PRINT) are preceded by a line number. However, the two terms are often used interchangeably. Don't let the students get distracted by precise memorization of vocabulary, but do, if possible, help them to use computer vocabulary correctly and consistently.

Chapter 8 Loading and Saving

STUDENT ASSIGNMENTS
Student Text pp. 29-37
Activity Worksheets (none)

VOCABULARY
CLOAD—The programming statement used to load data from the cassette tape recorder into ATARI's memory.

CSAVE—The programming statement used to save data from ATARI's memory through storage on a cassette tape recorder.

Filename—The name given to a program as it is saved to a storage device, such as the disk drive. The filename can be any combination of eight letters and numbers, but must begin with a letter.

LIST—The programming command that tells ATARI to display on the screen any program in memory.

LOAD D: FILENAME—The programming statement used to load data from the disk drive into ATARI's memory.

SAVE D: FILENAME—The programming statement used to save data from ATARI's memory to the disk drive.

OBJECTIVES
a. The student will learn to save and load a program with the aid of a storage device—a cassette tape recorder or a disk drive system.
b. The student will learn to use the LIST command to display programs in ATARI's memory.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students greatly extend their programming capabilities by learning to use storage devices—the cassette tape recorder and disk drive system. By having this storage capability, students can save their programs for later use and modification. Also, they will learn about the variety of commercial programs currently available.

Students also learn to use the LIST command to display programs in ATARI's memory.

Any quality cassette tape or diskette is acceptable for use with ATARI. Do not purchase "bargain brand" materials, which sometimes cause problems at the worst possible moment. Caution the students about proper care for the tapes or diskettes—to store them in proper containers, to keep them dust free, to avoid touching exposed surfaces, to use a felt tip pen and a "soft touch" if writing on a diskette label, and to keep the materials away from heat and magnetic sources.

Diskettes can store a tremendous amount of data—much more than will be needed by one student. If a single diskette is used by several students, caution them not to use the same filenames. Doing so will cause the second program to be stored at the same location as the first, which then will be erased. To prevent this problem, filenames can be identified on a written list accompanying the diskettes, or in the Disk Operating System (DOS). (Consult the disk drive manual on how to use DOS.)

Get a commercial program, or a program you have developed, and make it available on a cassette and/or diskette so students can practice loading and saving. If available, a short game program is fun for such practice. You may want to use the following number-guessing program.

```
  5 DIM B$(15),C$(20)
 10 ? "WHAT IS YOUR NAME";
 20 INPUT C$
 30 LET N=INT (100*RND(1)+1)
 40 ? " [ESC] [SHIFT] [CLEAR <]
 50 REM **BEGIN GAME**
 60 ? "WELL∅";C$;"∅THIS IS A GUESS A NUMBER GAME."
 70 FOR T=1 TO 2000:NEXT T
 80 ? [ESC] [SHIFT] [CLEAR <]
 90 ? "GUESS A NUMBER BETWEEN 1 AND 100"
100 INPUT G
110 IF G=N THEN 140
120 IF G>N THEN ? "HIGH, TRY AGAIN":GOTO 100
130 IF G<N THEN ? "LOW, TRY AGAIN":GOTO 100
140 REM ** CORRECT GUESS **
145 FOR T=1 TO 500:NEXT T
150 ? "YOU ARE FANTASTIC,∅";C$;"!"
170 ? :?
180 ? "YOU GUESSED IT!"
190 FOR T=1 TO 1000:NEXT T
200 ? [ESC] [SHIFT] [CLEAR <]
210 ? "WANT TO PLAY AGAIN∅"; C$;"?∅YES OR NO";
220 INPUT B$
230 IF B$="YES" THEN 30
240 ? :?
250 ? "THANKS FOR PLAYING, GOOD BYE FOR NOW∅";C$;"!"
260 END
```

Chapter 9 Teaching ATARI to Do Your Homework

STUDENT ASSIGNMENTS
Student Text p. 38
Activity Worksheets (none)

VOCABULARY
(none)

OBJECTIVE
The student will learn BASIC symbols for performing six types of arithmetic.

OVERVIEW AND NOTES TO THE TEACHER
The students are given an overview of the kind of arithmetic they will be doing with ATARI—addition, subtraction, multiplication, division, powers and square roots. The difference between the letter "O" and zero "0" is shown.

If you feel that the concepts of square root and powers are too difficult for some of your students, advice them that they may want to bypass those examples and problems.

In this chapter you may want to discuss the limitations of computers. Note that when powers are used, older ATARI models do not give the correct answer, but are slightly off (e.g., $2 \wedge 2 = 3.9999998$). This is a limitation of the ATARI computer that is corrected with newer models. (Oh, well. We all make mistakes! This limitation is discussed in Chapter 9 of the student text.)

Chapter 10 ATARI as a Calculator

STUDENT ASSIGNMENTS
Student Text p. 39
Activity Worksheets (none)

VOCABULARY
Direct or immediate mode—A state of computer operation in which a statement is executed immediately (e.g., PRINT 5+6 would immediately print 11 after the ⌷RETURN⌷ key is pressed). This contrasts with a delayed or program mode in which instructions are not executed until a program is run.

OBJECTIVE
The student will learn how to solve arithmetic equations in direct/immediate mode.

OVERVIEW AND NOTES TO THE TEACHER
The students are shown how ATARI can work as a calculator (a very expensive one), by placing arithmetic problems after a PRINT statement. PRINT may be abbreviated with a question mark (?).

The majority of *An ATARI for Kids* uses the computer in a programmed or delayed mode, whereby statements are grouped together in a program, and execution is delayed until the entire program is run. In this chapter, students will be working with the computer in the immediate or direct mode. The computer will carry out an operation immediately after the ⌷RETURN⌷ key is pressed.

STUDENT ASSIGNMENTS
Student Text pp. 40-41
Activity Worksheets pp. 18-25 (Programmer's Pastime #1, #2, #3, #4; Component #2 FUN PAGE; Evaluate Yourself)

VOCABULARY
(none)

OBJECTIVES
a.   The student will understand the order in which ATARI performs arithmetic.

b.   The student will learn how to write a PRINT statement in an abbreviated form.

OVERVIEW AND NOTES TO THE TEACHER

Emphasize to the students the importance of learning programming short-cuts so that their programs will be efficient as well as effective. Although not necessary at this point, memory conservation becomes important as more sophisticated programming is learned.

In remembering the order that ATARI performs arithmetic, the following mnemonic may be of help to some students:

Please	Excuse	My	Dear	Aunt	Sally
A	X	U	I	D	U
R	P	L	V	D	B
E	O	T	I	I	T
N	N	I	S	T	R
T	E	P	I	I	A
H	N	L	O	O	C
E	T	I	N	N	T
S	S	C			I
E		A			O
S		T			N
		I			
		O			
		N			

Arithmetic processes inside parentheses are always done first. Exponents or powers are done second. Multiplication and division are joined together in the above mnemonic because whichever is on the left will be done before the other. The same is true for addition and subtraction, whichever is on the left will be done first.

If powers and/or square roots are not appropriate for your students, advise them to bypass those problems.

Chapter 12 What Else Can ATARI Do for Me?

STUDENT ASSIGNMENTS

Student Text p. 46

Activity Worksheets (none)

VOCABULARY

(none)

OBJECTIVE

The student will understand that ATARI can perform operations besides arithmetic.

OVERVIEW AND NOTES TO THE TEACHER

The students are briefly introduced to other things that ATARI can do besides arithmetic.

Chapter 13 Flow Diagramming

STUDENT ASSIGNMENTS
Student Text pp. 47-50
Activity Worksheets pp. 26-30 (Programmer's Pastime #5, #6, #7)

VOCABULARY
Algorithm—A step-by-step method used to solve a problem.
Flow chart—A diagram which shows all the steps of an algorithm in the correct order.
Flow diagramming—The process of illustrating program components in a clear, step-by-step fashion.
Processing box—the rectangular shaped box in a flow chart that represents "something to be done".

OBJECTIVES
a.   The student will understand the necessity of solving a problem through utilization of a step-by-step algorithm.
b.   The student will understand how a flow chart uses an algorithm and is a pre-step to writing successful programs.

OVERVIEW AND NOTES TO THE TEACHER
The students are introduced to the importance of solving problems and writing programs in a clear, step-by-step fashion. The flow chart is used to teach logical, sequential programming.

Help the student understand there is often no single correct way to solve algorithms or diagram flow charts. Often steps can be modified, reversed, or substituted.

Chapter 14 More About Flow Charts

STUDENT ASSIGNMENTS
Student Text pp. 53-54
Activity Worksheets pp. 31-34 (Programmer's Pastime #8, #9, #10)

VOCABULARY
Decision box—The diamond-shaped box in a flow chart that represents a decision to be made.
Single-alternative decision step—A situation in a flow chart in which there is one "detour" from a decision box.

OBJECTIVE
The student will understand how a single-alternative decision step works in a flow chart.

OVERVIEW AND NOTES TO THE TEACHER
The students are introduced to the flow chart box that represents making decisions with one alternative.
Remember, the steps in a flow chart often can be ordered in several ways. The steps in solving algorithms or diagramming flow charts can be modified, reversed, or substituted.

Chapter 15 Double Detours

STUDENT ASSIGNMENTS
Student Text pp. 55-56
Activity Worksheets pp. 35-37 (Programmer's Pastime #11, #12)

VOCABULARY
Double-alternative decision step—A situation in a flow chart in which there are two "detours" from a decision box.

OBJECTIVE
The student will understand how a double-alternative decision step works in a flow chart.

OVERVIEW AND NOTES TO THE TEACHER
The students are introduced to the flow chart boxes that represents making decisions with two alternatives—if "yes," do one thing, if "no," do another thing.

Chapter 16 Loop De Loop

STUDENT ASSIGNMENTS

Student Text pp. 59-60

Activity Worksheets pp. 38-39 (Programmer's Pastime #13, #14)

VOCABULARY

Loop—A program situation, represented by an arrow in a flow chart, in which a certain step is repeated over and over.

OBJECTIVE

The student will understand that a loop can be used to repeat certain steps in a program.

OVERVIEW AND NOTES TO THE TEACHER

One of the goals of programming is to write efficient and appropriate programs. In this chapter, students are introduced to looping, an important technique in producing efficient programs.

Help students understand that programming often includes an ongoing process of revision. Encourage students to constantly evaluate and improve the programs they write.

Chapter 17 Putting it All Together

STUDENT ASSIGNMENTS
Student Text pp. 61-64
Activity Worksheets pp. 40-44 (Programmer's Pastime #15, #16)

VOCABULARY
Immediate or direct mode—An operational state of the computer in which statements typed on the screen are executed immediately when the RETURN is pressed. These statements do not have line numbers.

Program or delayed mode—An operational state of the computer in which statements typed on the screen are placed in the computer's memory when RETURN is pressed. These statements must have line numbers, and are stored in memory as part of a program until ATARI is given the RUN command.

OBJECTIVE
The student will learn how to take the algorithm steps from a flow chart and code them in a BASIC program.

OVERVIEW AND NOTES TO THE TEACHER
This chapter discusses the programming step of moving from the flow chart to the actual program written in BASIC. Two important BASIC commands—GOTO, which directs the program to a specified line, and RUN, which causes the program to be executed, are discussed.

Note on Programmer's Pastime #16 that some of the examples can be programmed correctly in two ways (e.g., 20 ? 100*10−1000 or 20 ? "0"). Familiarize your students with the idea that programs often can be written several different ways.

Chapter 18 Printing Whole Equations

STUDENT ASSIGNMENTS
Student Text pp. 67-68
Activity Worksheets p. 45 (Programmer's Pastime #17)

VOCABULARY
(none)

OBJECTIVE
The student will learn how to program the computer to print whole equations.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students work with printing an equation by placing it inside quotation marks (e.g., 20 ? "4+5="), and printing the calculated answer to an equation (e.g., 30 ? 4+5).

Chapter 19 A Different Way

STUDENT ASSIGNMENTS
Student Text pp. 69-71
Activity Worksheets pp. 46-50 (Programmer's Pastime #18; Component 3 Fun Page; and Evaluate Yourself)

VOCABULARY
Print zones or fields—Areas on the screen in which information is printed.

OBJECTIVES
a.   The student will learn how to use commas and semi-colons to print equations on one screen line.
b.   The student will understand the layout of the print zones on the screen and how a comma or semi-colon affects the output.

OVERVIEW AND NOTES TO THE TEACHER
This chapter introduces a shortcut for writing more efficient programs. Students learn how to use commas and semi-colons to print equations on one screen line. (Students should be encouraged to use writing shortcuts whenever possible.)

Blank spaces (∅) become important inside the quotation marks in order to get the proper spacing in the output. Sometimes it is difficult for students to remember to use blanks. This will come with practice.

Chapter 20 ATARI's Memory

VOCABULARY
Address—The location at which information is stored in the memory.
LET—The program statement which assigns a value to a variable.
Variable—A quantity that is capable of changing (varying) its value (e.g., X=1, X=2, X=3. "X" is a variable. Its value changes from 1 to 2 to 3, or to any other value assigned to it).

OBJECTIVES
a. The student will understand the basics of how ATARI's memory operates.
b. The student will learn what a numeric variable is, and how to properly name it.
c. The student will learn how to use the LET statement to assign a value to a variable address.

OVERVIEW AND NOTES TO THE TEACHER
Memory, the part of the ATARI computer which stores information, is introduced to the students. Each memory cell can be thought of as an electronic mailbox. Each mailbox can be assigned a name or address, and one piece of information can be stored at each address. The information at each address can be changed and is therefore called a variable.

The variables discussed in this chapter deal only with numbers, and therefore are called numeric variables. Later, students will learn about alphanumeric or string variables which deal with letters, numbers, and other characters.

The LET statement tells the computer what to call a given memory mailbox, and what contents (value) to assign to that mailbox. For example:

10 LET X=56   X   address
               - - -
               56   contents
                     (value)

Chapter 21 Using Variables

STUDENT ASSIGNMENTS
Student Text pp. 76-77
Activity Worksheets pp. 52-53 (Programmer's Pastime #20, #21)

VOCABULARY
Data—Information

OBJECTIVES
a.   The student will understand the difference between the address and the contents of a variable.
b.   The student will learn how to use variables in a program to print both variable names and variable contents.

OVERVIEW AND NOTES TO THE TEACHER
Variables are an important part of working with the ATARI. They allow a programmer to store information and then refer back to it and use it later.
The students learn how to command ATARI to PRINT both variable names, and variable contents.

Chapter 22 Using Variables in Equations

STUDENT ASSIGNMENTS
Student Text pp. 79-81
Activity Worksheets pp. 54-55 (Programmer's Pastime #22, #23)

VOCABULARY
(none)

OBJECTIVE
The student will learn how to use variables in a program to print and solve arithmetic equations.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students practice solving and printing arithmetic equations by using programs. They see what variables can be assigned number values, or values of other variables, or values of equations.

Notice the use of semicolons in the following line:

$$30 ? A; \text{``} \emptyset + \emptyset \text{''}; B; \text{``} \emptyset = \emptyset \text{''}; A+B$$

It is OK to mix variables (A and B) with symbols for arithmetic processes (+ and =), but they must be separated by semicolons. If students forget to do so, ATARI will give an error message.

Chapter 23 Important Information

STUDENT ASSIGNMENTS
Student Text pp. 82-83
Activity Worksheets pp. 56-60 (Programmer's Pastime #24, #25, #26)

VOCABULARY
(none)

OBJECTIVE
The student will understand the ramifications of using a LET statement before a PRINT statement in a program, and vice versa.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students practice using PRINT and LET statements in the correct order.

Chapter 24 A Shortcut

STUDENT ASSIGNMENTS
Student Text p. 86
Activity Worksheets pp. 61-62 (Programmer's Pastime #27)

VOCABULARY
(none)

OBJECTIVE
The student will learn how to use colons and commas to shorten a program.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are given more techniques for shortening programs and thus making them more efficient. The particular focus here is on using colons to combine LET statements, and commas to combine PRINT statements.

It is possible to omit "LET" when writing LET statements. It is up to the individual to do so or not. Using "LET" is often clearer for beginning programmers, so it is used in the programs throughout *An ATARI for Kids*.

Chapter 25 — What Type of Numbers Does ATARI Like?

STUDENT ASSIGNMENTS

Student Text pp. 87-88

Activity Worksheets pp. 63-65 (Programmer's Pastime #28; Component 4 Fun Page, Evaluate Yourself)

VOCABULARY

E (Exponential) notation or floating point notation—A way of representing very large or very small numbers.

OBJECTIVES

a. The student will understand what type of numbers ATARI can work with.

b. The student will understand how to read E notations (floating point notations.)

OVERVIEW AND NOTES TO THE TEACHER

In this chapter, students are shown some problems of working with fractions, and are introduced to a notation system used by the computer for very large numbers.

There is a problem that may arise when using large numbers. When 9 digits are used, ATARI will switch to E Notation. However, when 10 digits are used, ATARI will accurately represent the first nine, but will change the last digit to a zero.

Chapter 26 FOR-NEXT Looping

STUDENT ASSIGNMENTS
Student Text pp. 90-93
Activity Worksheets pp. 66-73 (Programmer's Pastime #29, #30, #31)

VOCABULARY
Counter-controlled loop—A programming loop that can be executed for a specified number of times.
FOR-NEXT—A program statement that allows counter-controlled loops to be made.

OBJECTIVE
The student will learn how to use FOR-NEXT loops in a program to create counter-controlled looping.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are shown another way to create loops within programs. This type of loops involves a "counter" so that it can be executed a specified number of times.

Chapter 27 Stepping

STUDENT ASSIGNMENTS
Student Text pp. 96-98
Activity Worksheets pp. 74-78 (Programmer's Pastime #32, #33)

VOCABULARY
STEP—A program statement that allows counter-controlled loops to be counted in a certain pattern (e.g., by fives, tens, twenties, etc.)

OBJECTIVE
The student will learn how to use the STEP statement in a program to make ATARI count in number patterns.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are shown how to use the STEP statement to make ATARI count in number patterns—by fives, tens, etc.

Chapter 28 A Counter

STUDENT ASSIGNMENTS
Student Text pp. 99-101
Activity Worksheets pp. 79-80 (Programmer's Pastime #34)

VOCABULARY
Counter—A program technique used to keep track of the number of times a loop has been executed.

OBJECTIVE
The student will learn how to use a counter in a program to keep track of how many times a loop has been executed.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are shown how to use a counter to keep track of how many times a loop has been executed.

Chapter 29 A Clean Trick

STUDENT ASSIGNMENTS
Student Text pp. 102-105
Activity Worksheets pp. 81-85 (Programmer's Pastime #35, #36)

VOCABULARY
FOR-NEXT time loop—A loop using a FOR-NEXT statement that causes a pause in the printing of output on the screen.

OBJECTIVES
a.   The student will learn how to program ATARI to clear the screen during the execution of a program.
b.   The student will learn how to use a FOR-NEXT time loop to slow down the printing of output on the screen.
c.   The student will learn how to use colons to place all of a FOR-NEXT statement on a single line.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are shown techniques for clearing the screen during a program and slowing down the printing of output on the screen. They also are given a shortcut for writing FOR-NEXT statements on a single line.

Chapter 30 Blinkers

STUDENT ASSIGNMENTS
Student Text p. 106
Activity Worksheets pp. 86-88 (Programmer's Pastime 37)

VOCABULARY
(none)

OBJECTIVE
The student will learn how to program blinking output.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are shown how to use FOR-NEXT time loops to make output blink on and off ATARI's screen.

Chapter 31 Special Commands

STUDENT ASSIGNMENTS
Student Text pp. 109-111
Activity Worksheets (none)

VOCABULARY
BYE—The programming command that allows BASIC to be exited and puts ATARI in the memo pad mode (older models), or the testing mode (XL models).
CONT—The programming command that allows ATARI to continue after it has been stopped because the BREAK key has been pressed.
Memo pad mode—The operating mode that ATARI enters after the BYE command has been given.
STOP—The programming statement used on a numbered line within a program to stop a program run.

OBJECTIVES
a.   The student will learn how to use the BYE and CONT commands.
b.   The student will learn how to use the STOP statement.
c.   The student will learn the function of the memo pad mode.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students learn to use the BYE and CONT commands, and the STOP statement. They also are introduced to the memo pad and testing modes. The older Atari models have the memo pad mode, which can be very helpful for beginning programmers. It allows them to leave BASIC and experiment with the keyboard without changing any program in ATARI's memory. The XL models have a testing mode, which allows the user to check how well ATARI's memory, keyboard and audiovisual systems are working.

Chapter 32 Debugging

STUDENT ASSIGNMENTS
Student Text pp. 112-113
Activity Worksheets pp. 89-90 (Component 5 Fun Page and Evaluate Yourself)

VOCABULARY
Bugs—Mistakes that a programmer can make while writing a program.
Debugging—The process of getting rid of program mistakes.

OBJECTIVES
a. The student will understand the technique of debugging.
b. The student will understand three basic types of computer errors.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students consider techniques to correct mistakes in their programs.

Chapter 33 Strings

STUDENT ASSIGNMENTS
Student Text pp. 115-116
Activity Worksheets pp. 91-92 (Programmer's Pastime #38, #39)

VOCABULARY
Dimension (DIM)—A program statement used to reserve a specified number of characters to be used with string variables.
String or alphanumeric variable—A variable that consists of letters, numbers, or special characters (<, =, $, etc.).

OBJECTIVES
a.   The student will understand what a string variable is and the correct way to use it in a program.
b.   The student will learn how to dimension a string variable.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are introduced to string or alphanumeric variables, which can store letters, numbers, special characters (<, =, $, etc.), words or whole sentences.

Tell the students that the word "alphanumeric" gives a clue that it is both an alphabetic and numeric variable. This contrasts with the numeric variable (employing only numbers) which have been used to this point.

Dimensioning string variables often causes errors from beginning programmers. Either they use a string variable without previously using a DIM statement, or they do not "reserve" enough spaces with the statement.

Chapter 34 Input

STUDENT ASSIGNMENTS
Student Text pp. 117-120
Activity Worksheets pp. 93-98 (Programmer's Pastime #40, #41)

VOCABULARY
INPUT—A program statement that allows data to be typed into the program while the program is running.

Interactive program—A program that allows input to be typed into the program while it is running.

OBJECTIVES
a. The student will understand what interactive programming is.
b. The student will learn how to use the INPUT statement in a program to interact with the computer.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are taught to interact with programs by using INPUT statements.

Students may need to be reminded of the difference between "INPUT" as a program statement, and "input" as a general term for data that goes into the computer.

Chapter 35 IF-THEN

STUDENT ASSIGNMENTS
Student Text pp. 121-126
Activity Worksheets pp. 99-106 (Programmer's Pastime #42, #43, #44, #45)

VOCABULARY
Complement—Using the opposite of a question or a sign. (e.g., < is the complement of >).
IF-THEN—A program statement used to make comparisons and establish conditional situations.

OBJECTIVES
a.   The student will learn the BASIC signs used to make comparisons and how to use them in a program with the IF-THEN statement.
b.   The student will understand how to use the complement of a question in a program which uses the IF-THEN statement.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are introduced to IF-THEN, a powerful programming statement. Using IF-THEN, comparisons can be made and conditional happenings can be established. Students also consider the complement (opposite) of an IF-THEN statement.

Chapter 36 Alphabetizing

STUDENT ASSIGNMENTS
Student Text pp. 128-129
Activity Worksheets p. 107-108 (Programmer's Pastime #46)

VOCABULARY
(none)

OBJECTIVES
a.   The student will understand how ATARI compares letters to alphabetize them.
b.   The student will learn how to program ATARI to alphabetize words.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are introduced to techniques for alphabetizing letters and words.

Chapter 37 Remarks

STUDENT ASSIGNMENTS
Student Text pp. 130-131
Activity Worksheets pp. 109-111 (Programmer's Pastime #47)

VOCABULARY
Documentation—Using remark statements to note and clarify what is happening in a program.
Remark (REM)—A program statement used to place clarifying notes throughout a program. Remarks are not executed as part of the program.
Style—Using a variety of techniques to develop easy-to-read programs.

OBJECTIVE
The student will learn how to document a program effecitvely by using remark (REM) statements.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students use remark statements to develop a more effective programming style.

Help students realize that as they write longer and longer programs, a good, easy-to-read programming style is extremely important.

Chapter 38 READ-DATA

STUDENT ASSIGNMENTS
Student Text pp. 132-138
Activity Worksheets pp. 112-121 (Programmer's Pastime #48, #49, #50, #51, #52)

VOCABULARY
Dummy Data—Data that is read as a signal that the READ-DATA pointer is at the end of the DATA list.

Pointer—An electronic device that marks the location of data being read from a DATA list.

READ-DATA—Two programming statements that work together making it possible to place data in a program as it is typed on the keyboard.

OBJECTIVE
The student will learn how to utilize the READ-DATA statements in a program.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students use READ-DATA statements to place data in a program directly from the keyboard.

READ-DATA are extremely powerful programming statements because they allow previously written programs to be easily modified by changing the DATA listing.

Chapter 39 Problem-Solving Programming

STUDENT ASSIGNMENTS
Student Text pp. 140-145
Activity Worksheets pp. 122-135 (Programmer's Pastime #53; Component 6 Fun Page; Evaluate Yourself)

VOCABULARY
(none)

OBJECTIVE
The student will learn a process of programming geared to solving problems.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students bring all of their programming skills together to write programs to solve problems.

Although computers can do a wide variety of things to make our lives easier, one of the most valuable uses of computers is to solve problems. The skill of writing good problem-solving programs will require a lot of practice.

Some students will want to bypass some of the suggested seven steps for writing problem-solving programs. Encourage them, particularly at first, to use all the steps. Later on, some students can modify or develop a different system that works better for them.

Chapter 40 Conversions

STUDENT ASSIGNMENTS

Student Text pp. 149-150

Activity Worksheets pp. 136-147 (Programmer's Pastime #54, #55, #56)

VOCABULARY

Conversion—Changing one type of information to another type of information.

Conversion equation—The program equation used to convert one type of information to another.

OBJECTIVE

The student will learn how to write a conversion program.

OVERVIEW AND NOTES TO THE TEACHER

In this chapter, students learn to write programs that allow one type of information to be changed to another type—e.g., feet to inches, miles to kilometers, etc.

Chapter 41 Random Numbers and Integers

STUDENT ASSIGNMENTS
Student Text pp. 152-155
Activity Worksheets pp. 148-156 (Programmer's Pastime #57, #58, #59, #60, #61, #62, #63)

VOCABULARY
Computer-Assisted Instruction (CAI)—Computers used for teaching/learning purposes.

Function—Certain operations that are done automatically, like a built-in small program.

INT—The program function used to create whole numbers, or integers, in a program.

Integers—Whole numbers; without fractions or decimals.

Random Numbers—List of numbers that are in no particular order.

RND—The program function used to create random numbers in a program.

OBJECTIVE
The student will learn how to use the RND and INT functions in a program to accomplish certain tasks.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students learn to write programs that use the random number (RND) and integer (INT) functions.

The formula for creating random integers is somewhat confusing because of the great number of parentheses being used. Help students to remember that the innermost set of parentheses is always done first.

Also, the INT function is somewhat confusing when used with negative numbers. This function always rounds the number DOWN to the nearest integer. Therefore −5.3 is rounded down to −6, −14.1 to −15, etc.

Chapter 42 Making Sounds and Music

STUDENT ASSIGNMENTS
Student Text pp. 156-159
Activity Worksheets pp. 157-159 (Programmer's Pastime #64, #65)

VOCABULARY
SOUND (SO.)—The programming statement used with ATARI to make sounds and music.

OBJECTIVE
The student will learn how to use the SOUND statement to program ATARI to make sounds and music.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students learn to program ATARI for sounds and music.

This is an enjoyable chapter for students, for they learn techniques that allow them to creatively add music and sounds to their programs. Allow students considerable time to experiment with ATARI's sounds. This can cause a classroom disturbance, however, unless some rules are established. In particular, students should keep the loudness variable soft enough so as not to bother others.

In the student text, the section dealing with TONE asks why there is a pause between each of the eight tones when the program is executed. This is because tones are produced only with the even numbers between 0 and 14. As the program is written, line 10 causes ATARI to play all tone values between 0 and 14 (FOR S=0 TO 14). The odd numbers will not work, and therefore there will be pauses. This could be changed by modifying line 10 to read: FOR S=0 TO 14 STEP 2.

Chapter 43 Graphics

STUDENT ASSIGNMENTS
Student Text pp.160-168
Activity Worksheets pp. 160-166 (Programmer's Pastime #66, #67, #68)

VOCABULARY
COLOR (C.)—The programming statement that allows the color of graphics to be changed.
DRAWTO (DR.) The programming statement used to draw lines in the graphics mode.
GRAPHICS (GR.)—The programming statement that places ATARI in the graphics mode.
Graphics mode—The state of operation in which graphics can be produced.
PLOT (PL.)—The programming statement that allows a point to be placed at a specified location.
Text window—The lower four lines of the screen in graphics mode that allows text to be displayed with graphics.
X-coordinate—The first number in a PLOT statement. It specifies the column position.
Y-coordinate—The second number in a PLOT statement. It specifies the row position.

OBJECTIVE
The student will learn how to use the GRAPHICS, COLOR, PLOT, and DRAWTO statements to produce simple graphics.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students learn to program ATARI for simple graphics using the GRAPHICS, COLOR, PLOT, and DRAWTO statements.

Students often need considerable assistance and practice in locating points by the X and Y-coordinates. The graph pages used in the Activity Workbook Programmer's Pastime #66 may be useful. These pages can be used as masters for making transparencies for large group work, or for individual student copies. It is often useful to laminate the individual copies. Marked with water soluble pens, the graphs can then be reused many times.

Valuable grid coordinate practice can be provided to some younger students by using the floor tile in an open space for a graph. A variety of games can be developed that require students to move to specified locations.

Chapter 44 More Graphics

STUDENT ASSIGNMENTS

Student Text pp. 170-173
Activity Worksheets pp. 167-168 (Programmer's Pastime #69, #70)

VOCABULARY

SETCOLOR (SE.)—The programming statement that allows the color of the screen, text window and graphics to be changed.

OBJECTIVES

a.   The student will learn how to use the SETCOLOR statement to produce some advanced graphics.
b.   The student will learn to create a graphics screen without the text window.
c.   The student will learn to combine both graphics and sound in a single program.

OVERVIEW AND NOTES TO THE TEACHER

In this chapter, students learn some advanced graphics techniques using the SETCOLOR statement, and a way to eliminate the text window from the graphics screen. They will also combine graphics and sound together in a single program.

SETCOLOR is difficult and frustrating for some students to use because it has so many variations when used in combination with the other graphics statements. Remind them that, taken slowly and with a lot of practice, SETCOLOR becomes a very useful and creative tool which produces a lot of variations. As they experiment with SETCOLOR, sugguest that they vary only one factor at a time to see what happens.

Chapter 45 Writing Game Programs

STUDENT ASSIGNMENTS
Student Text pp. 174-177
Activity Worksheets pp. 169-177 (Programmer's Pastime #71)

VOCABULARY
User-friendly—A program that is easy to use.

OBJECTIVES
a. The student will understand the three basic types of computer games.
b. The student will learn how to write a game program that is user-friendly.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students learn to write computer game programs.

Chapter 46 You are a Creative Programmer!

STUDENT ASSIGNMENTS
Student Text p. 178
Activity Worksheets pp. 178-179 (Component 7 Fun Page; Evaluate Yourself)

VOCABULARY
(none)

OBJECTIVE
The student will understand that programming a computer can be a stimulating, creative experience.

OVERVIEW AND NOTES TO THE TEACHER
In this chapter, students are challenged to become creative computer programmers.

EXPLORING ATARI'S KEYBOARD #1

Answers

Take a few minutes to explore ATARI's keyboard. Press single keys, and keys while SHIFT and CTRL are being held down, and see what happens. (The keys on the far right row are for commercial programs. Do not press them.)

Finish drawing the key or keys that must be pressed to get ATARI to type what is shown on the screen at the right. Check your answers by using ATARI.

1. [A]   ⌐A

2. [# / 3] [# / 3]   ⌐33

3. [SHIFT] [# / 3]   ⌐#

4. [CTRL] [X]   ⌐⊥

5. [CTRL] [D]   ⌐◄

6. [V]   ⌐V

7. [% / 5] [& / 6]   ⌐56

8. [CLEAR / <]   ⌐<

9. [SHIFT] [:]   ⌐:

10. [:]   ⌐;

EXPLORING ATARI'S KEYBOARD #2

Answers

1. Turn ATARI on.

2. Press [RETURN]
   What did the cursor do? <u>Moved down</u>
   <u>one line</u>

3. Press [SPACE] (It's the long bar at the bottom.
   It's not labeled.)
   What did the cursor do this time? <u>Moved</u>
   <u>right one space</u>

4. Press [SHIFT] and [CLEAR <] together.
   What did the cursor do? <u>Screen was</u>
   <u>cleared, cursor moves to upper left</u>
   <u>(home) position</u>

5. Type your name (first, middle, and last).

6. Hold [CTRL] and press [←+\] five times.
   Which way did the cursor move? <u>Left</u>
   Did anything happen to the writing on the
   screen? <u>No</u>
   This is how you can move the cursor around
   the screen without erasing any of the writing.

7. Hold [CTRL] and press the various cursor control
   keys (those with arrows [↑−] [↓=] [←+\] [→*^]).
   Notice how the cursor moves around the
   screen without changing the writing.

8. Use the [CTRL] and cursor control keys (with ar-
   rows) to move the cursor to the first letter of
   your middle name.

9. Press the space bar two times.
   What happened? <u>Two letters are erased</u>
   <u>(to the right)</u>

10. Press the [DELETE BACKS] key five times.
    What happened? <u>5 letters and spaces</u>
    <u>are erased (to the left)</u>

11. Hold [SHIFT] while pressing [CLEAR <] .
    What happened? <u>Screen is cleared, cursor</u>
    <u>moves to upper left (home) position</u>

EXPLORING ATARI'S KEYBOARD #3

Answers

Finish drawing the key (or keys) that must be pressed to get ATARI to perform each special function.

1. | CTRL | | ↑‒ | Move the cursor up.

2. | SHIFT | | CLEAR < | Clear the screen and send the cursor home.

3. | CTRL | | →* ^ | Move the cursor right.

4. | ЛЬ | or | ◢ | Begin reverse field printing.

5. | ЛЬ | or | ◢ | End reverse field printing.

6. | CTRL | | ←\ + | Move the cursor left.

7. | CTRL | | DELETE BACK S | Delete a letter.
   or | DELETE BACK S |

8. | SHIFT | | DELETE BACK S | Delete a line.

9. | CTRL | | INSERT > | Insert a space.

10. | SHIFT | | INSERT > | Insert a line.

EXPLORING ATARI'S KEYBOARD #4

Answers

1. Turn ATARI on.

2. Type I LIKE YOU ATARI.

3. Press [DELETE BACKS] five times.
   What happened? <u>Five letters erased</u>
   <u>(to the left)</u>

4. Hold [SHIFT] and press [INSERT >] five times.
   What happened? <u>Five lines are added</u>

5. Hold [SHIFT] and press [DELETE BACKS] five times.
   What happened? <u>Five lines are erased</u>

6. Hold [SHIFT] and press [CLEAR <].
   What happened? <u>All writing erased,</u>
   <u>cursor moves to upper left (home) position</u>

7. Type I LOVE YOU ATARI.

8. Hold [CTRL] and press [-\ +] until the cursor is on
   the Y of YOU.

9. Hold [CTRL] and press [DELETE BACKS] four times.
   What happened? <u>Three letters and</u>
   <u>a space are erased (to the right)</u>

10. Hold [CTRL] and press [INSERT >] four times.
    What happened? <u>Four spaces are added</u>

11. Type YOU back in the new space.

12. If you have time, try typing some lines of your
    own, and use the various keys to do some
    screen editing.

EXPLORING ATARI'S KEYBOARD #5

Answers

Quick Review

Tell what happens when these keys are pressed:

1. [SHIFT] [CLEAR <]   Screen is cleared, cursor moves to upper left

2. [SHIFT] [INSERT >]   Lines are added

3. [SHIFT] [DELETE BACK S]   Lines are erased (deleted)

4. [CTRL] [INSERT >]   Spaces are added

5. [CTRL] [DELETE BACK S]   Characters are erased (deleted)

6. [CTRL] [↑ −]   Cursor moves upward

7. [CTRL] [↓ =]   Cursor moves downward

8. [CTRL] [← \ +]   Cursor moves left

9. [CTRL] [→ ^ *]   Cursor moves right

10. [(SPACE)]   Cursor moves to right, adds blank space

11. [DELETE BACK S]   Cursor moves left, characters are deleted

EXPLORING ATARI'S KEYBOARD #6

Mine the Diamonds

created by Wendy Cheldelin

1. Turn ATARI on.

2. Clear the screen.

3. Press ⌈CTRL⌉ and hold it down. Press the ⌈P⌉ key 5 times.

4. Press ⌈CTRL⌉ and hold it down. Now press the ⌈I⌉ key once. On you screen should be 5 "rocks" and 1 "diamond."

5. On the same line, make 4 to 5 more rocks followed by 1 diamond until the line is filled up and the cursor has moved to the line below.

6. Fill one more screen line with rocks and diamonds.

7. Now the challenge begins! Mine the diamonds by erasing all of the rocks. Be careful! Don't erase any diamonds.

Answers will vary

EXPLORING ATARI'S
KEYBOARD #7

Answers

Design Your Own Game!

1. Design a game that requires some understanding of ATARI's keyboard. Write clear directions on how to play your game in the space below.

2. Ask a friend to play your game!

Answers will vary

COMPONENT 1 FUN PAGE

Answers

A completed crossword puzzle with the following answers:

- 7 Across: GRAPHICS
- 8 Across: KEYBOARD
- 13 Across: CURSOR
- 9 Across: SCREEN
- 14 Across: REC
- 15 Across: MEMORY
- 10 Across: SPACE
- 11 Across: BYTE
- 12 Across: INSERT
- 2 Down: MICROCOMPUTER
- 1 Down: BASIC
- 4 Down: RETURN
- 6 Down: PROGRAM
- 5 Down: SHIFT
- 3 Down: DELETE
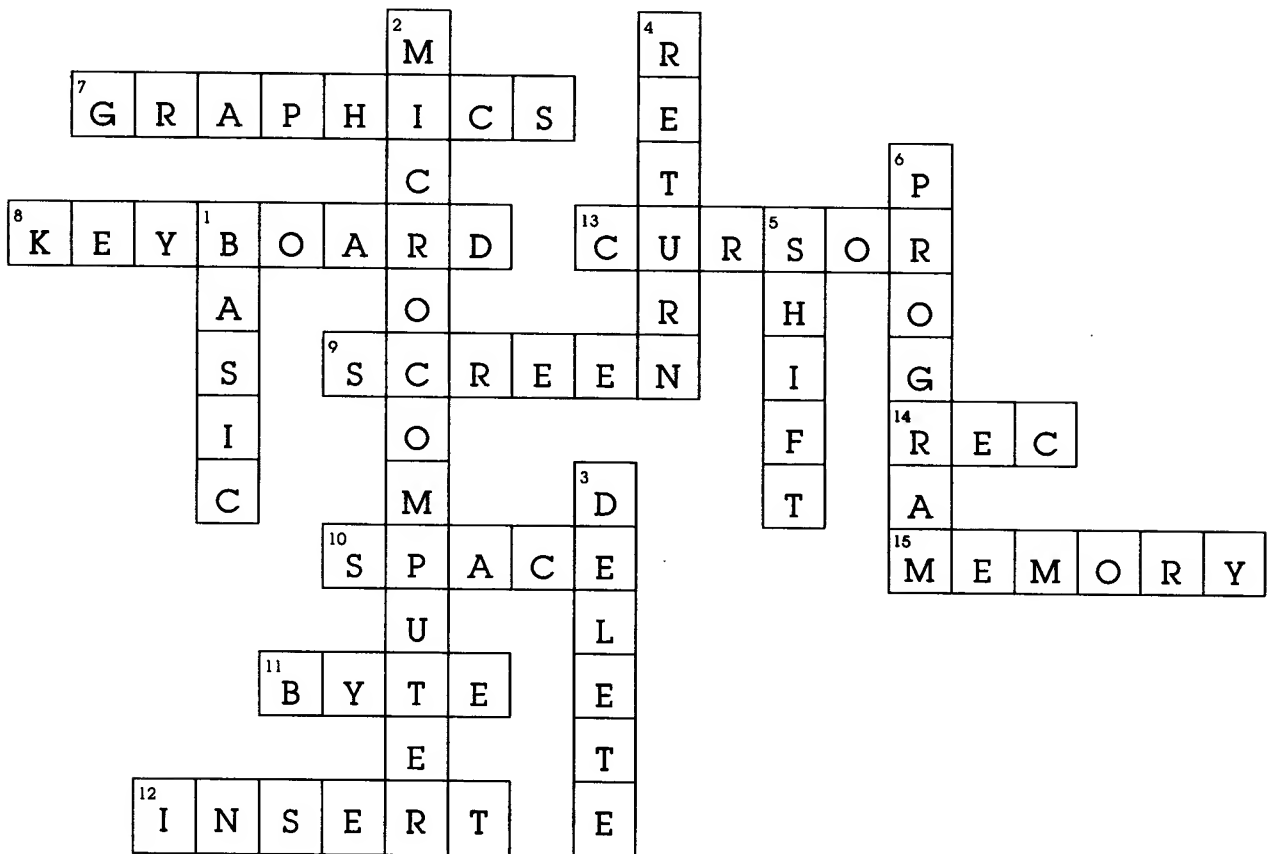
Speak

1. Write a program that tells ATARI to print:

   COMPUTER PROGRAMMING IS FUN!

2. Make sure each line of your program begins with a line number.

3. Check your program to make sure there are no mistakes.

4. RUN your program on the computer.

Use this format

```
10 PRINT "            "
20 END
```

Write your program here

```
10 PRINT "COMPUTER PROGRAMMING IS
   FUN!"
20 END
```

Speaking Nonstop

1. Write a program that tells ATARI to print your name over and over again!
2. RUN your program on ATARI.

Use this format

```
10 PRINT "              "
     (Type your name inside the quotes.)
20 GOTO 10
```

Write your program here

```
10 PRINT "(NAME)"
20 GOTO 10
```

Top-Secret

1. Your mission is to write a program that tells ATARI to print a top-secret message in a secret code. Use the graphic symbols on the keys as your code. For example, if we wanted a word in our message to say SAW, the code would be + ⊢ ⊤ because these graphic symbols appear on the S, A, and W keys.
2. Give your program to a friend. Have your friend RUN it on ATARI and try to decode the secret message.

Your success as a secret agent depends on this program. Good luck! (This page will self-destruct in two days if your program is not finished.)

Write your program here

```
10 PRINT "(CODE IN GRAPHICS)"
20 END

        Answers will vary
```

Computer Art

1. Write a program that tells ATARI to print a design using graphic symbols.
2. Make your design be printed over and over on the screen.
3. RUN the program on ATARI.

Write your program here

```
10 PRINT "(DESIGN USING GRAPHICS)"
20 GOTO 10

        Answers will vary
```
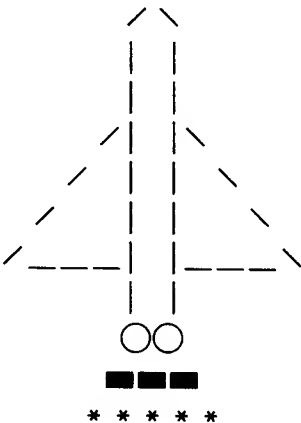
Answers

Shuttle Launch

1. You can write a program that will tell ATARI to make moving pictures by using these five things:
   line numbers
   PRINT statements
   quotation marks
   GOTO statement
   graphic symbols
2. Use the format below to create a program that launches a rocket.
3. RUN the program on ATARI.

Use this format

```
 10 PRINT  "        / \        "
 20 PRINT  "        | |        "
 30 PRINT  "        | |        "
 40 PRINT  "      / | | \      "
 50 PRINT  "     /  | |  \     "
 60 PRINT  "    /   | |   \    "
 70 PRINT  "   /____| |____\   "
 80 PRINT  "        | |        "
 90 PRINT  "        OO         "
100 PRINT  "      ■ ■ ■        "
110 PRINT  "     * * * * *     "
120 PRINT  "                   "
130 PRINT  "                   "
140 GOTO 10
```

NOTE: It is important to leave lines 120 and 130 blank inside the quotation marks so the rockets are spaced out when they move on the screen.

Write your program here

Answers will vary

You may have more than 140 lines in your program.

PROGRAMMER'S PASTIME #1

Answers

Write the symbol that ATARI uses for each arithmetic operation.

addition	+	multiplication	*
powers	∧	subtraction	−
division	/	square root	SQR ( )

How would you type each equation to get answers from ATARI?

1.	$457+99\times6$	? 457+99*6
2.	$\sqrt{64}$	? SQR (64)
3.	$26\div2^2$	? 26/2∧2
4.	$777\times555\div222$	? 777*555/222
5.	8^3-16	? 8∧3−16
6.	$\sqrt{22}\div88$	? SQR (22)/88
7.	$\sqrt{49}+765$	? SQR (49)+765
8.	$98+88\times66\div2^4$	? 98+88*66/2∧4

Show how you would type the equations above using only one PRINT (?) statement.

? 457+99*6, SQR (64), 26/2∧2, 777*555/222,

8∧3−16, SQR (22)/88, SQR (49)+765,

98+88*66/2∧4

PROGRAMMER'S PASTIME #2

What is the order that ATARI does arithmetic in equations of many numbers?

__Parentheses_____are done first.

__Powers_____are done second.

__Multiplication__ and __division_____ are done third (left to right).

__Addition_____ and __subtraction____ are done last (left to right).

Use your mental powers and write the answers that ATARI would give for these equations. Remember to do the arithmetic in the same order that ATARI would!

1. 2*3+1   __7__
2. 2+8*2*1   __18__
3. 3*3+9+20   __38__
4. 11+4*3   __23__
5. 22+8+12*1   __42__

Try some more:

1. 8/2-3   __1__
2. 20/4+6-5   __6__
3. 30-10/2   __25__
4. 50-20/10+3   __51__
5. 16/2-8/2   __4__
6. 14/2+4/2-6   __3__

PROGRAMMER'S PASTIME #3

Use your mental powers and write the answer the computer would give for these equations:

1. 4*4+6/2   _19_
2. 8/4+3*3   _11_
3. 20−(4+5*2)+15   _21_
4. 6+14/7*5   _16_
5. (9/3−2)*(7*2+4)   _18_
6. (7+2−4+6*1)/1   _11_

POWER!

Powers are also called **EXPONENTS**. Read the following examples and figure out how powers work on your own:

1. 10∧1=10*1=10
2. 10∧2=10*10=100
3. 10∧3=10*10*10=1000

Try some more:

1. 2∧1=2*1=2
2. 2∧2=2*2=4
3. 2∧3=2*2*2=8
4. 2∧4=2*2*2*2=16

Use your mental powers and solve the powers by filling in the blanks:

1. 3∧1= _3*1_   = _3_
2. 3∧2= _3*3_   = _9_
3. 3∧3= _3*3*3_   = _27_
4. 3∧4= _3*3*3*3_   = _81_

Now try these:

1. 4∧1= _4*1_   = _4_
2. 4∧2= _4*4_   = _16_
3. 4∧3= _4*4*4_   = _64_
4. 4∧4= _4*4*4*4_   = _256_

CHALLENGE

1. 2∧4*5	80
2. 5*2∧4	80
3. 7+3∧2	16
4. 5∧2−13	12
5. 100/10∧2	1
6. 12∧2−5*10	94
7. 8∧2*(4+5)	576
8. 200−(6+3∧3)+7	174

(REMEMBER—If you check these on the computer, ATARI gives a slightly inaccurate answer when working with powers.)

In each of the following equations, put a 1 under the part the computer would do first, a 2 under the second part it would do, and so on.

Example: 82+72 / 2−(4+22)+9 ∧ 3
$$\quad\quad\quad\quad \underline{4}\quad\underline{3}\;\underline{5}\quad\underline{1}\quad\;\underline{6}\;\underline{2}$$

1. 3000−(13−6*4)+8 ∧ 3+9
$$\quad\quad\underline{4}\quad\underline{2}\;\underline{1}\quad\underline{5}\;\underline{3}\;\underline{6}$$

2. 900 / 7 ∧ 2+(16−9 ∧ 4) ∧ 3
$$\quad\quad\underline{5}\;\underline{3}\;\underline{6}\quad\underline{2}\;\underline{1}\quad\;\underline{4}$$

PROGRAMMER'S PASTIME #4

Cowboy Clyde typed in the following equations, but ATARI wouldn't give him answers. Do you know why?

Find out what's wrong with the way his equations are typed. Write the correct way in the blanks.

1. ? 62+4×20   ? 62+4*20
2. 23/4×6   ? 23/4*6
3. ? SQR 16   ? SQR (16)
4. 80/4+2×3×SQR 25 ? 80/4+2*3*SQR (25)

CHALLENGE

If you want certain arithmetic in a long equation that you want to be done FIRST, you should put parentheses around it. (ATARI always does what's in parentheses first.) Let's say you want 4+3*2 to equal 14. If you type: 4+3*2 ATARI will give you 10 because multiplication is done before addition. So it becomes 4+6, which equals 10.

If you want 4+3*2 to equal 14, you must use parentheses like this: (4+3)*2.

$$7*2=14$$

Rewrite each equation below and put parentheses around what ATARI should do first in order to make the equation TRUE. HINT: You might have to use *two* sets of parentheses for some equations.

Example: $7*3+2=35$ $7*(3+2)=35$

1. $9/2+1=3$ $9/(2+1)=3$
2. $6*2+2=24$ $6*(2+2)=24$
3. $3\wedge3-1=9$ $3\wedge(3-1)=9$
4. $4+2-1*5=9$ $4+(2-1)*5=9$
5. $12-3+6/3=9$ $12-(3+6)/3=9$
6. $20-10\wedge4+2=10002$ $(20-10)\wedge4+2=10002$
7. $12/4+2=2$ $12/(4+2)=2$
8. $9-5\wedge2=16$ $(9-5)\wedge2=16$
9. $50+10/18+10+2=2$ $(50+10)/(18+10+2)=2$
*10. $10+4-7\wedge2*1+3-3=193$ $(10+4-7)\wedge2*(1+3)-3=193$

* means it's an extra tough problem!

COMPONENT 2 FUN PAGE

Answers
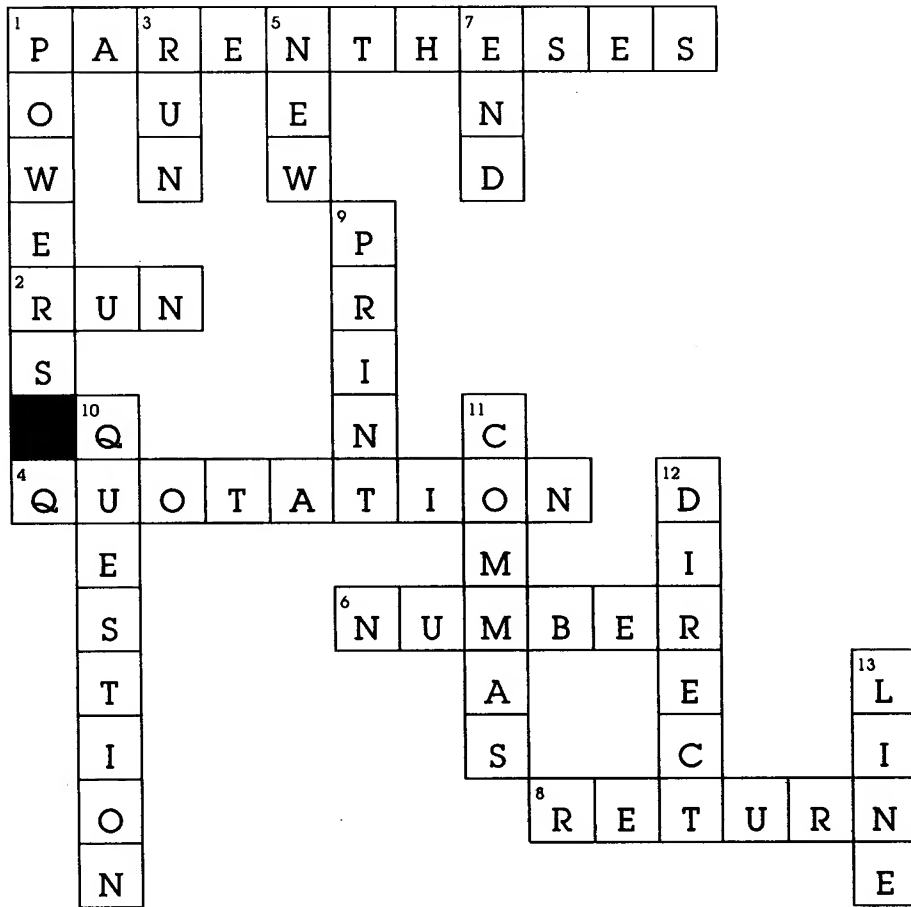
```
P A R E N T H E S E S
O   U   E       N
W   N   W       D
E       9P
2R U N   R
S       I
 10Q    N  11C
4Q U O T A T I O N  12D
     E      M      I
     S  6N U M B E R
     T      A      E  13L
     I      S      C   I
     O  8R E T U R N
     N              E
```

Answers

For each flow chart, fill in the blank boxes with the step you think would fit. Make sure your steps are in the right order.

Algorithm/Flow Chart #1
How to feed your pet elephant.

```
   ( START )
       ↓
┌─────────────┐
│ Get out the │
│ peanuts     │
└─────────────┘
       ↓
┌─────────────┐
│ Put the     │
│ peanuts in  │
│ the bowl    │
└─────────────┘
       ↓
┌─────────────┐
│ Call your   │
│ pet elephant│
└─────────────┘
       ↓
   ( STOP )
```
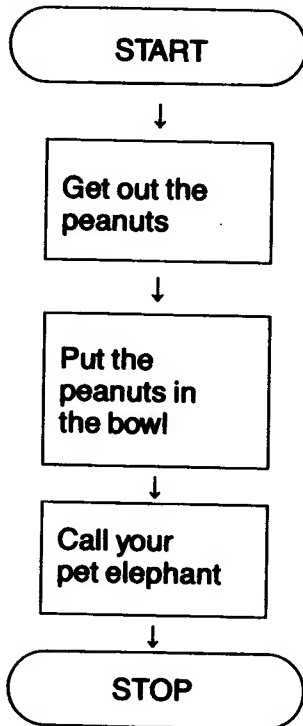
Missing Steps

STOP

Call your pet elephant.

START

Get out the peanuts.

Algorithm/Flow Chart #2
How to wash your pet skunk.
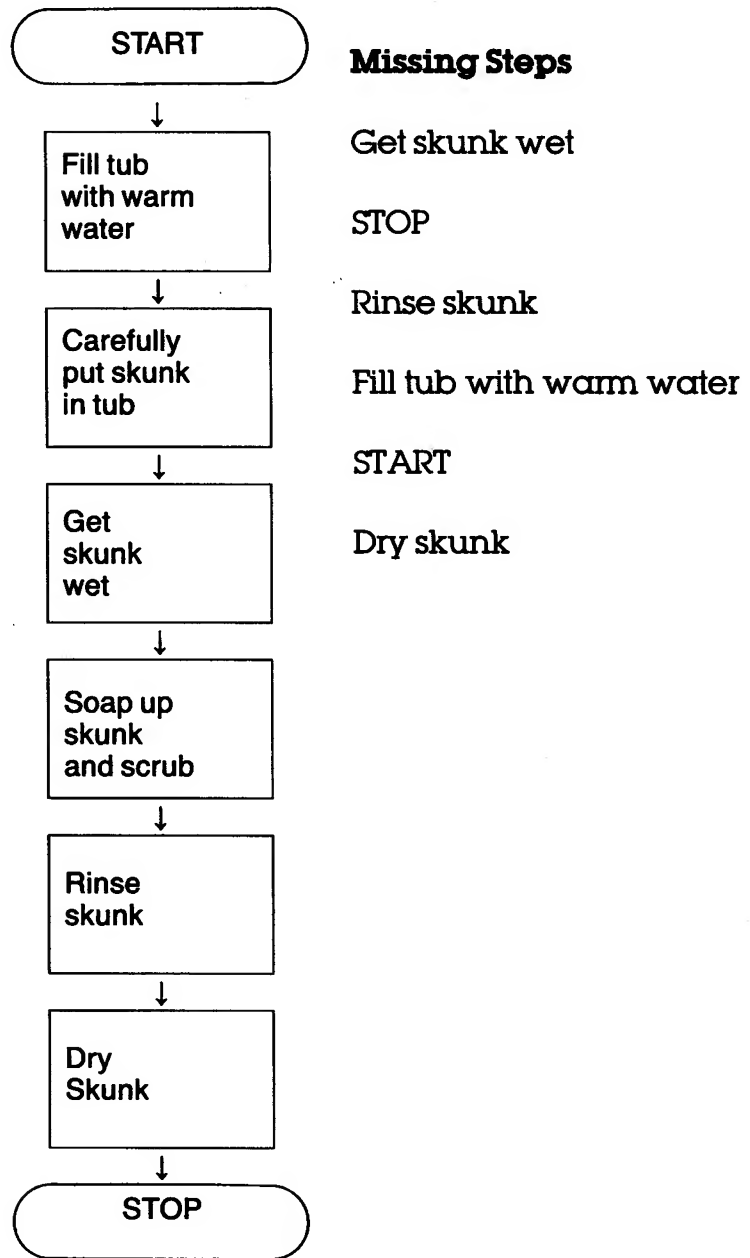
```
      ( START )

         ↓
  ┌──────────────┐
  │ Fill tub     │
  │ with warm    │
  │ water        │
  └──────────────┘
         ↓
  ┌──────────────┐
  │ Carefully    │
  │ put skunk    │
  │ in tub       │
  └──────────────┘
         ↓
  ┌──────────────┐
  │ Get          │
  │ skunk        │
  │ wet          │
  └──────────────┘
         ↓
  ┌──────────────┐
  │ Soap up      │
  │ skunk        │
  │ and scrub    │
  └──────────────┘
         ↓
  ┌──────────────┐
  │ Rinse        │
  │ skunk        │
  └──────────────┘
         ↓
  ┌──────────────┐
  │ Dry          │
  │ Skunk        │
  └──────────────┘
         ↓
      ( STOP )
```

Missing Steps

Get skunk wet

STOP

Rinse skunk

Fill tub with warm water

START

Dry skunk

PROGRAMMER'S PASTIME #6

For each flow chart, fill in the blank boxes with the step you think would fit. Make sure your steps are in the right order.

Algorithm/Flow Chart #1
How to walk your pet alligator.

START

↓

Muzzle
the
alligator

↓

Put on
his
collar

↓

Attach
leash to
collar

↓

Go for
a walk

↓
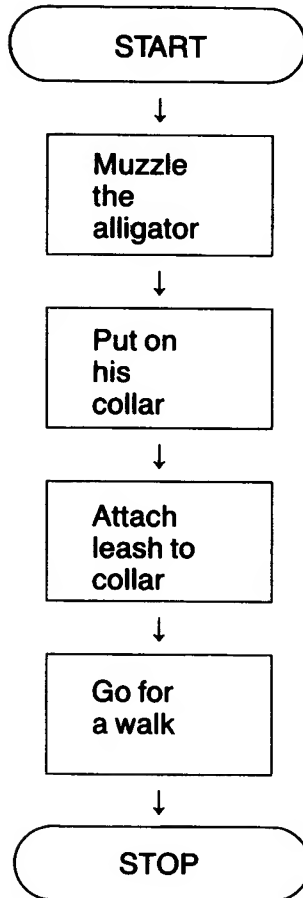
STOP

Missing Steps

Attach leash to collar

Go for a walk

START

Muzzle the alligator

STOP

Put on his collar

Algorithm/Flow Chart #2
How to make an ice cream cone.

```
        ┌─────────────┐
       (    START     )
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Get out     │
        │ a cone      │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Get out     │
        │ ice cream   │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Scoop out   │
        │ ice cream   │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Put scoop of│
        │ ice cream   │
        │ into cone   │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Pack ice    │
        │ cream firmly│
        │ into cone   │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Put away    │
        │ ice cream   │
        └─────────────┘
              ↓
        ┌─────────────┐
        │ Eat ice     │
        │ cream cone  │
        └─────────────┘
              ↓
        ┌─────────────┐
       (    STOP      )
        └─────────────┘
```
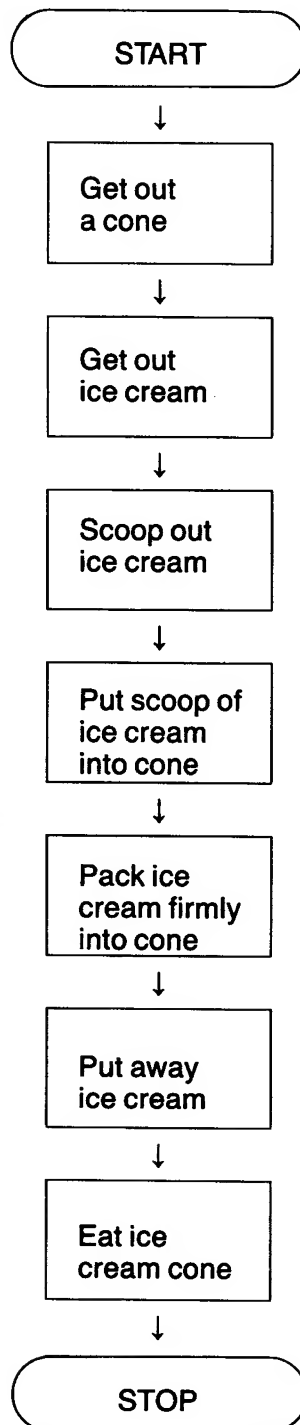
Missing Steps

Put away ice cream

Scoop out ice cream

START

Get out ice cream

STOP

Get out a cone

Put scoop of ice cream into cone

Pack the ice cream firmly into cone

Eat ice cream cone

Answers

Design an algorithm for how to make a peanut butter and banana sandwich. Write your algorithm in flow chart form.

The flow chart may look something like this:
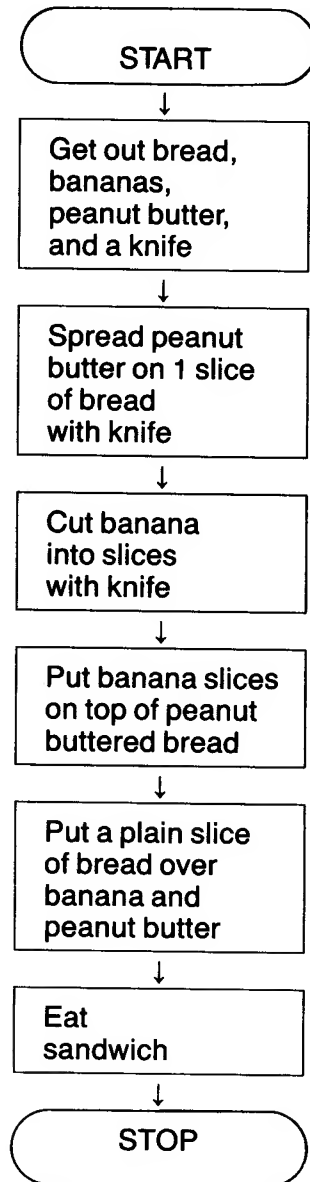
```
      ╭──────────────────╮
      │      START        │
      ╰──────────────────╯
               ↓
      ┌──────────────────┐
      │ Get out bread,    │
      │ bananas,          │
      │ peanut butter,    │
      │ and a knife       │
      └──────────────────┘
               ↓
      ┌──────────────────┐
      │ Spread peanut     │
      │ butter on 1 slice │
      │ of bread          │
      │ with knife        │
      └──────────────────┘
               ↓
      ┌──────────────────┐
      │ Cut banana        │
      │ into slices       │
      │ with knife        │
      └──────────────────┘
               ↓
      ┌──────────────────┐
      │ Put banana slices │
      │ on top of peanut  │
      │ buttered bread    │
      └──────────────────┘
               ↓
      ┌──────────────────┐
      │ Put a plain slice │
      │ of bread over     │
      │ banana and        │
      │ peanut butter     │
      └──────────────────┘
               ↓
      ┌──────────────────┐
      │ Eat               │
      │ sandwich          │
      └──────────────────┘
               ↓
      ╭──────────────────╮
      │      STOP         │
      ╰──────────────────╯
```

Answers

For each flow chart, fill in the blank boxes with the steps you think would fit. Make sure your steps are in the right order.

Algorithm/Flow Chart #1:
How to take a hot bath.
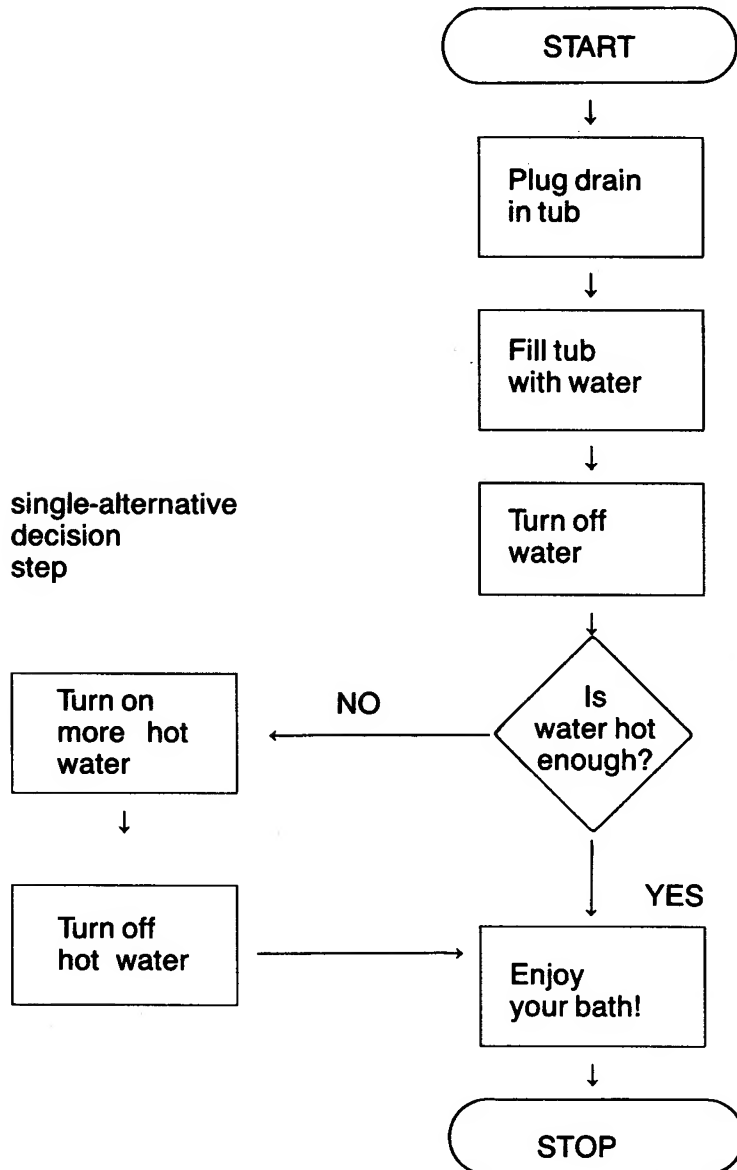
Missing Steps

Fill tub with water

Enjoy your bath

START

Turn off hot water

Turn off water

STOP

Is water hot enough?

Plug drain in tub

Turn on more hot water

single-alternative
decision
step

```
          ( START )
               ↓
        ┌─────────────┐
        │ Plug drain  │
        │ in tub      │
        └─────────────┘
               ↓
        ┌─────────────┐
        │ Fill tub    │
        │ with water  │
        └─────────────┘
               ↓
        ┌─────────────┐
        │ Turn off    │
        │ water       │
        └─────────────┘
               ↓
             ◇ Is
        water hot
        enough?
```

Turn on more hot water ← NO

Turn off hot water →

YES

Enjoy your bath!

↓

STOP

Algorithm/Flow Chart #2
How to eat a bowl of Rice Krispies.

This flow chart
has two
single-alternative
decision steps!
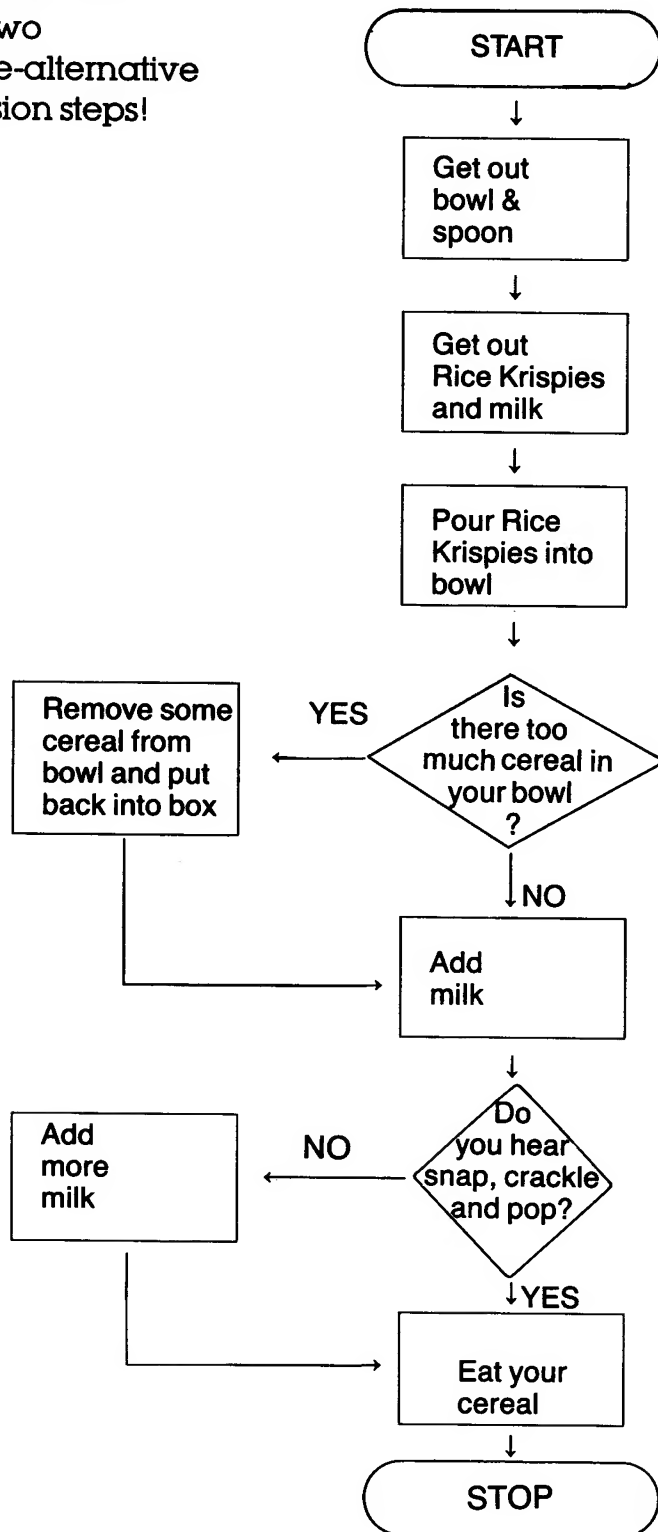
Missing Steps

Eat your cereal

Get out bowl & spoon

STOP

Add milk

Pour Rice Krispies into
bowl

Is there too much cereal in
your bowl?

Get out Rice Krispies and
milk

Remove some cereal from
bowl and put back into
box

START

```
                    ( START )
                        |
                        v
                  +-------------+
                  | Get out     |
                  | bowl &      |
                  | spoon       |
                  +-------------+
                        |
                        v
                  +-------------+
                  | Get out     |
                  | Rice Krispies|
                  | and milk    |
                  +-------------+
                        |
                        v
                  +-------------+
                  | Pour Rice   |
                  | Krispies into|
                  | bowl        |
                  +-------------+
                        |
                        v
+--------------+  YES  / Is there \
| Remove some  |<------| too much  |
| cereal from  |      | cereal in  |
| bowl and put |      | your bowl ?|
| back into box|       \         /
+--------------+           | NO
       |                   v
       |            +-------------+
       +----------->| Add         |
                    | milk        |
                    +-------------+
                        |
                        v
+--------------+  NO  / Do you hear\
| Add          |<-----| snap,crackle|
| more         |      | and pop?    |
| milk         |       \          /
+--------------+           | YES
       |                   v
       |            +-------------+
       +----------->| Eat your    |
                    | cereal      |
                    +-------------+
                        |
                        v
                    ( STOP )
```

PROGRAMMER'S PASTIME #9

Here are the steps of an algorithm to make a chocolate milkshake. Put the steps in order and make a flow chart. Be sure to show the SINGLE-ALTERNATIVE DECISION STEP.
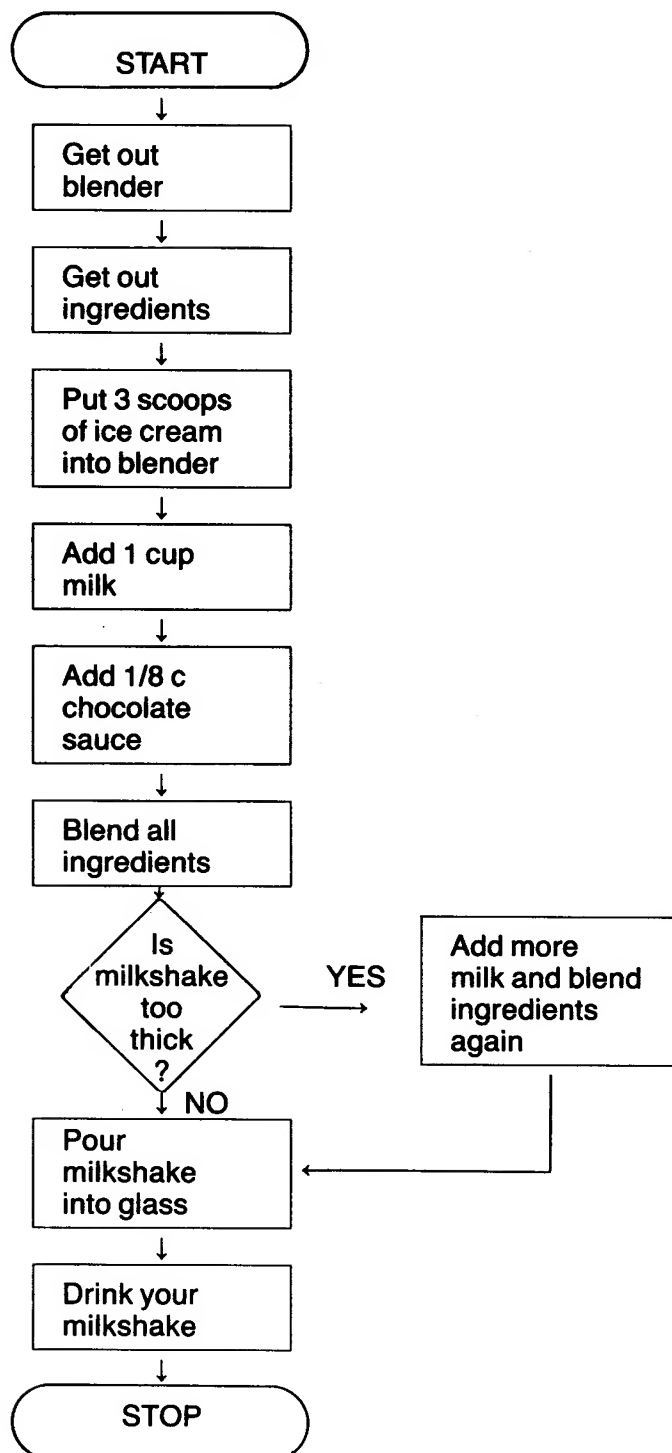
Steps

STOP

Add ⅛ cup chocolate sauce

Get out blender

Put 3 scoops of ice cream into blender

Is milkshake too thick?

Get out ingredients

Add 1 cup milk

Blend all ingredients

Add more milk and blend ingredients again

Pour milkshake into glass

START

Drink your milkshake

```
        ( START )
            ↓
    ┌─────────────┐
    │ Get out     │
    │ blender     │
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Get out     │
    │ ingredients │
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Put 3 scoops│
    │ of ice cream│
    │ into blender│
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Add 1 cup   │
    │ milk        │
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Add 1/8 c   │
    │ chocolate   │
    │ sauce       │
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Blend all   │
    │ ingredients │
    └─────────────┘
            ↓
       < Is         >        ┌─────────────┐
       < milkshake  > YES    │ Add more    │
       < too        > ────→  │ milk and blend│
       < thick?     >        │ ingredients │
            │ NO             │ again       │
            ↓                └─────────────┘
    ┌─────────────┐                │
    │ Pour        │ ←──────────────┘
    │ milkshake   │
    │ into glass  │
    └─────────────┘
            ↓
    ┌─────────────┐
    │ Drink your  │
    │ milkshake   │
    └─────────────┘
            ↓
        ( STOP )
```

80

PROGRAMMER'S PASTIME #10

Write an algorithm on how to fix yourself a cold glass of water. Make the algorithm into a flow chart. Your flow chart should have a SINGLE-ALTERNATIVE DECISION STEP.

Example: Ask the question, "IS THE WATER COLD ENOUGH?"

For the NO answer detour, your step might say: ADD ICE CUBES.
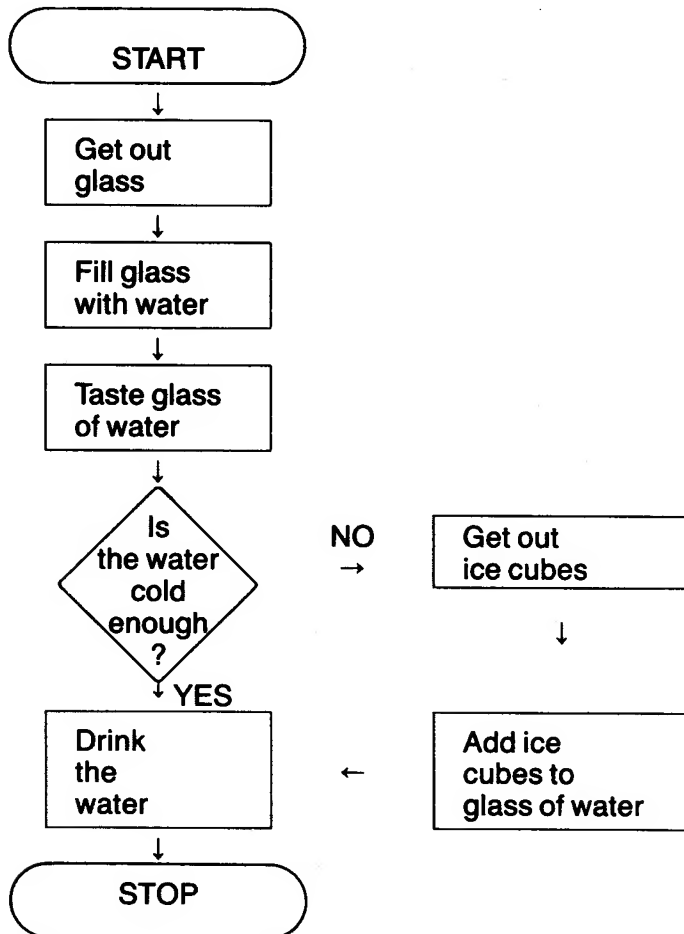
```
        ┌─────────────┐
        │    START    │
        └──────┬──────┘
               ↓
        ┌─────────────┐
        │ Get out     │
        │ glass       │
        └──────┬──────┘
               ↓
        ┌─────────────┐
        │ Fill glass  │
        │ with water  │
        └──────┬──────┘
               ↓
        ┌─────────────┐
        │ Taste glass │
        │ of water    │
        └──────┬──────┘
               ↓
             ◇ Is                        ┌─────────────┐
          the water    NO →              │ Get out     │
            cold                         │ ice cubes   │
           enough?                       └──────┬──────┘
              ↓ YES                             ↓
        ┌─────────────┐                  ┌─────────────┐
        │ Drink       │        ←         │ Add ice     │
        │ the         │                  │ cubes to    │
        │ water       │                  │ glass of    │
        └──────┬──────┘                  │ water       │
               ↓                         └─────────────┘
        ┌─────────────┐
        │    STOP      │
        └─────────────┘
```

Answers

For each flow chart, fill in the blank boxes with the steps you think would fit. Make sure your steps are in the right order.

Algorithm/Flow Chart: How to teach your pet bull to come when you call.

```
        ( START )
           ↓
   ┌──────────────────┐
   │ Climb into       │
   │ corral. Approach │
   │ bull carefully   │
   └──────────────────┘
           ↓
   ┌──────────────────┐
   │ Gently get       │
   │ bull's attention │
   │ by calling his   │
   │ name             │
   └──────────────────┘
           ↓
   ┌──────────────────┐
   │ Say "Come Toro"  │
   │ over and over    │
   └──────────────────┘
           ↓
   ┌──────────────────┐
   │ Hold out         │
   │ handful of hay   │
   └──────────────────┘
           ↓
```

```
┌──────────────┐        ◇ Is        ┌──────────────┐
│ Pet bull.    │  NO   the bull   YES │ Turn and run │
│ Quickly give │ ←──  charging   ──→ │ as fast      │
│ him the hay  │        at you?      │ as you can   │
└──────────────┘          ?          └──────────────┘
```

DOUBLE-ALTERNATIVE
DECISION STEP

```
   ┌──────────────┐
   │ Climb out    │
   │ of corral    │
   └──────────────┘
           ↓
        ( STOP )
```
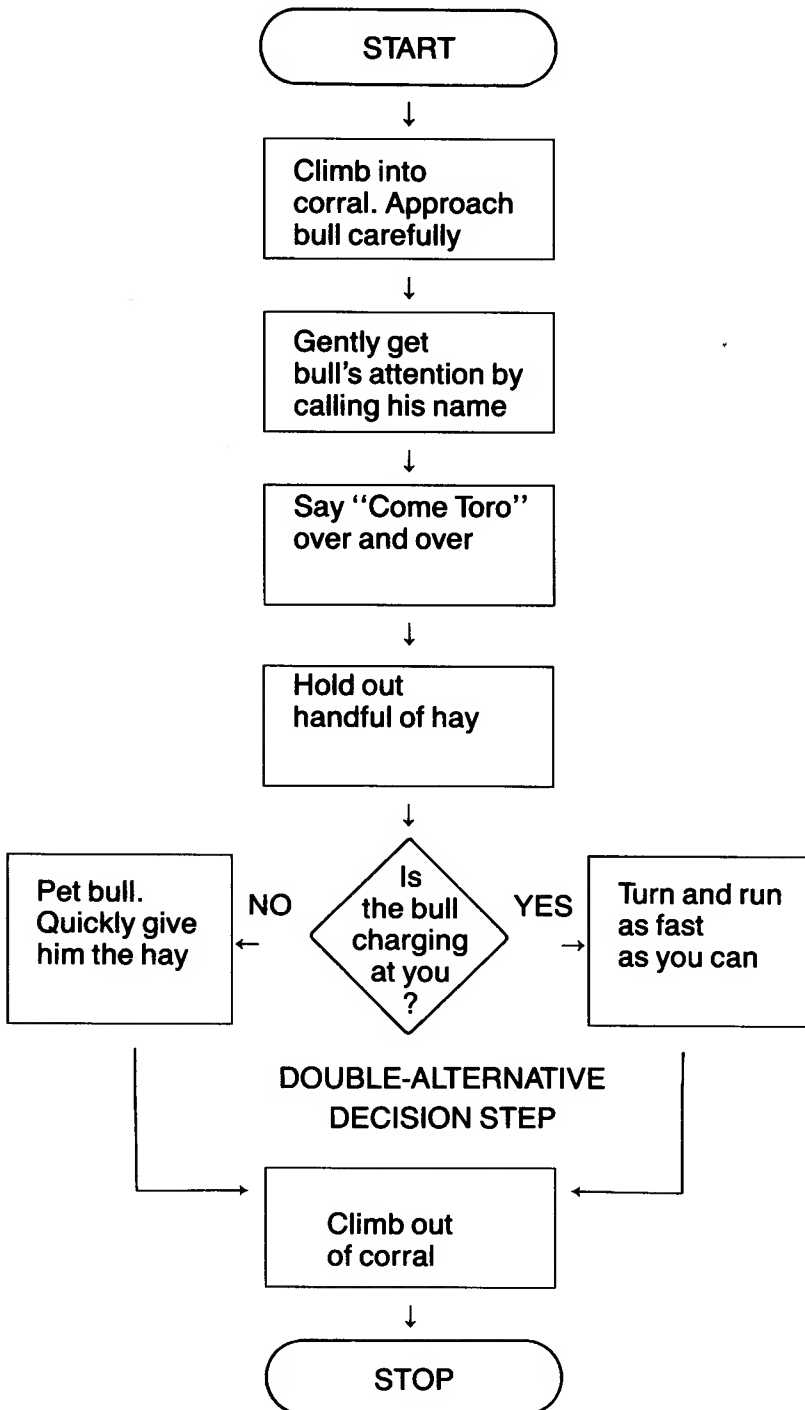
Missing Steps

Hold out handful of hay

Climb out of corral

START

Gently get bull's attention by calling his name

Pet bull. Quickly give him the hay

Say "COME TORO" over and over

Turn and run as fast as you can

STOP

Climb into corral. Approach bull carefully

Is the bull charging at you?

Algorithm/Flow Chart: How to bake cookies.

```
            ( START )
                |
                v
      +------------------+
      | Mix ingredients  |
      | according to     |
      | recipe           |
      +------------------+
                |
                v
      +------------------+
      | Put dough        |
      | on cookie        |
      | sheets           |
      +------------------+
                |
                v
      +------------------+
      | Put cookie       |
      | sheets in        |
      | hot oven         |
      +------------------+
                |
                v
      +------------------+
      | Wait 10 min.     |
      | then open        |
      | oven             |
      +------------------+
                |
                v
                /\
               /  \
+-----------+ / Are \ +-----------+
| Wait 3 min| <--NO / cookies \ YES--> | Turn      |
| Then turn |      \ done   /          | Off       |
| off oven  |       \  ?  /            | oven      |
+-----------+        \  /              +-----------+
      |               \/                     |
      |      DOUBLE-ALTERNATIVE              |
      |        DECISION STEP                 |
      |                                      |
      |      +------------------+            |
      +----->| Remove cookies   |<-----------+
             | from oven        |
             +------------------+
                     |
                     v
             +------------------+
             | Cool 5           |
             | minutes          |
             +------------------+
                     |
                     v
             +------------------+
             | Gobble           |
             | the cookies      |
             +------------------+
                     |
                     v
                 ( STOP )
```

Missing Steps

Wait 10 min. then open oven

STOP

Are cookies done?

Gobble the cookies

Mix ingredients according to recipe

Cool 5 minutes

Put cookie sheets in hot oven

Remove cookies from oven

Put dough on cookie sheet

START

PROGRAMMER'S PASTIME #12

Answers

Write an algorithm and flow chart on how to hit a baseball with a bat during a game when you are up to bat. Your flow chart must have a DOUBLE-ALTERNATIVE DECISION STEP.

Example: Question: IS THE BALL IN THE STRIKE ZONE?

YES	**NO**
KEEP YOUR EYE ON THE BALL	STEP AWAY FROM THE PLATE

Example

START
↓
Step up to home plate
↓
Position yourself to bat
↓
Wait for pitcher to pitch the ball
↓
Is ball in the strike zone ?

Step away from the plate ← NO

YES → Keep your eye on the ball
↓
Swing!
→ STOP

84
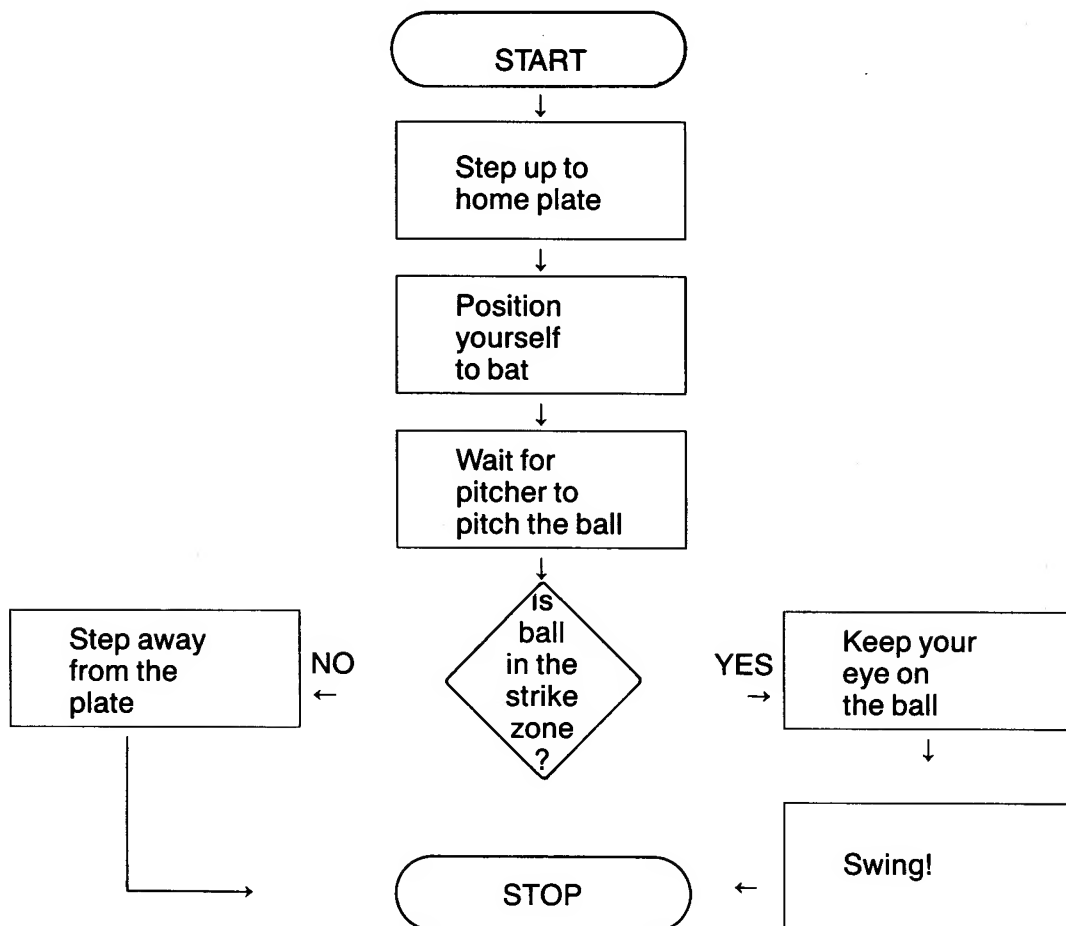
Answers

Fill in the blank boxes of the flow chart with the step you think should fit. Make sure the steps fit the loop.

Algorithm/Flow Chart: How to eat candy.

```
      ( START )
          ↓
    ┌───────────┐
    │ Unwrap    │
    │ candy     │
    └───────────┘
          ↓
    ┌───────────┐
    │ Take      │ ←──────┐
    │ a bite    │        │
    └───────────┘        │
          ↓          L   │
    ┌───────────┐    O   │
    │ Chew      │    O   │
    │ and       │    P   │
    │ swallow   │        │
    └───────────┘        │
          ↓              │
        ╱Is╲             │
      ╱ there ╲   YES    │
     ⟨more candy⟩────────┘
      ╲ left  ╱
        ╲ ? ╱
          ↓  NO
    ┌───────────┐
    │ Throw     │
    │ away      │
    │ wrapper   │
    └───────────┘
          ↓
    ┌───────────┐
    │ Brush     │
    │ your      │
    │ teeth     │
    └───────────┘
          ↓
      ( STOP )
```
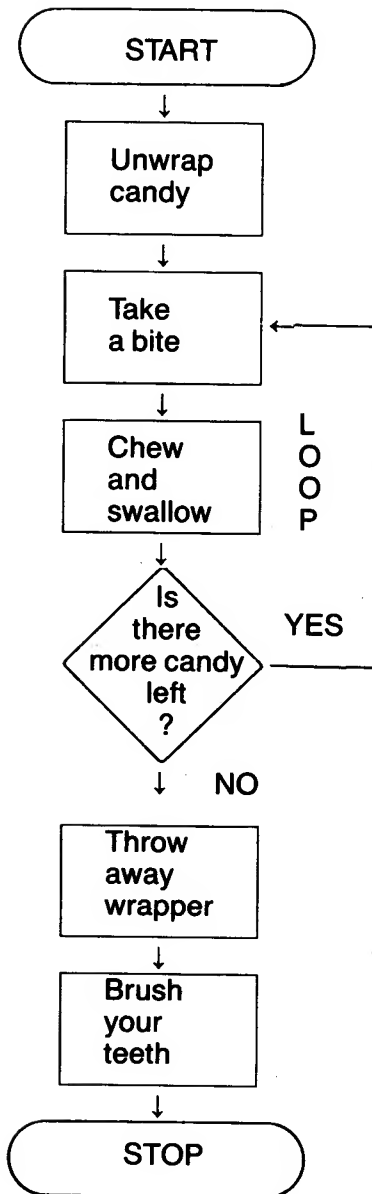
Missing Steps

Is there more candy left?

Brush your teeth

STOP

Throw away wrapper

Unwrap candy

START

Chew and swallow

Take a bite

PROGRAMMER'S PASTIME #14

Answers

Write an algorithm for how to wash your hair. Write the algorithm in flow chart form. Your flow chart should have a LOOP.

Example:

Question: IS THERE SOAP IN YOUR HAIR?
LOOP: YES → RINSE YOUR HAIR

Example

```
        ( START )
            ↓
    ┌───────────────┐
    │ Turn on       │
    │ warm          │
    │ water         │
    └───────────────┘
            ↓
    ┌───────────────┐
    │ Get hair      │
    │ wet           │
    └───────────────┘
            ↓
    ┌───────────────┐
    │ Apply         │
    │ shampoo.      │
    │ Lather up     │
    └───────────────┘
            ↓
    ┌───────────────┐
    │ Rinse         │ ←──┐
    │ hair          │    │
    └───────────────┘    │
            ↓            │
          ╱ Is ╲         │
         ╱ there ╲ YES   │
        ╱ soap in ╲──────┘
        ╲  your  ╱
         ╲ hair ╱
          ╲  ? ╱
            ↓  NO
    ┌───────────────┐
    │ Dry your      │
    │ hair          │
    └───────────────┘
            ↓
        ( STOP )
```

86
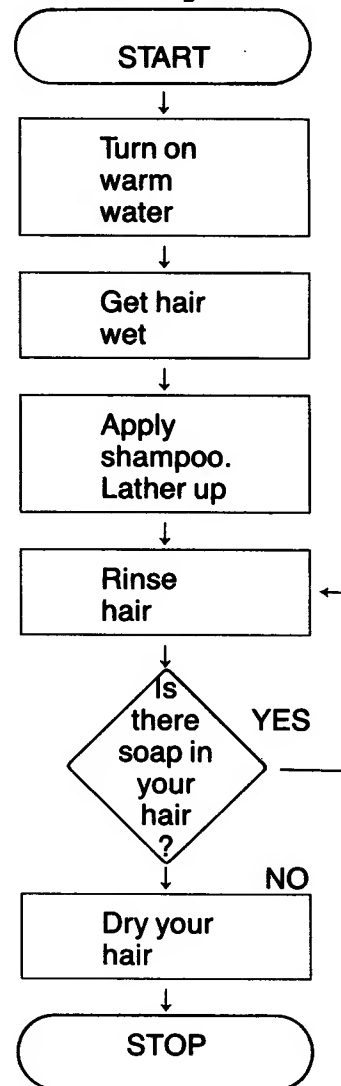
For each algorithm/flow chart, write a program in BASIC. (HINT — Pressing [CTRL] and [⋮] will give a tree-like graphic.)

1. (START)

Print
"SAVE OUR
TREES"

PRINT
"♠ ♠ ♠"

```
10 ? "SAVE OUR
      TREES"
20 ? "♠ ♠ ♠"
30 GOTO 10
```

2. (START)

PRINT
"My favorite
friend is"

PRINT
"You!"

```
10 ? "MY FAVORITE
      FRIEND IS"
20 ? "YOU!"
30 GOTO 20
```

3.
```
START
  ↓
Print
"4*400="
  ↓
Print
4*400
  ↓
STOP
```

```
10 ?"4*400="
20 ?4*400
30 END
```

4.
```
START
  ↓
Print
"10∧2="
  ↓
Print
10∧2
  ↓
Print
"10∧3="
  ↓
Print
10∧3
  ↓
STOP
```

```
10 ?"10∧2="
20 ?10∧2
30 ?"10∧3="
40 ?10∧3
50 END
```
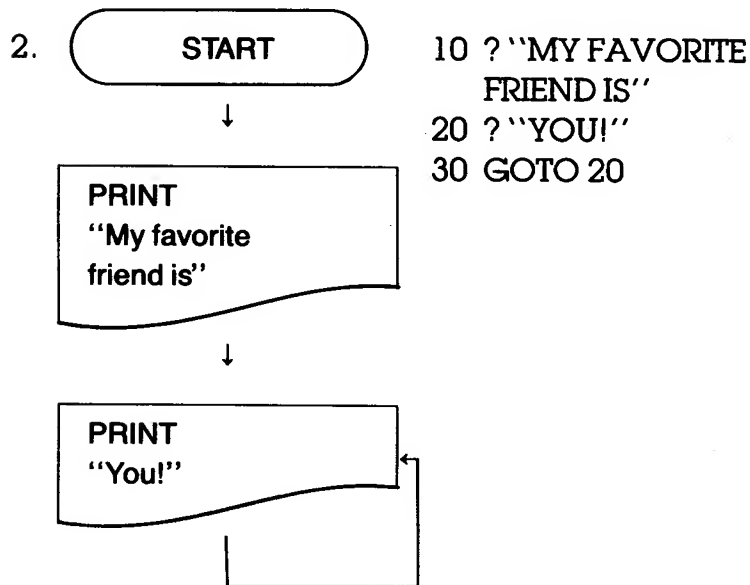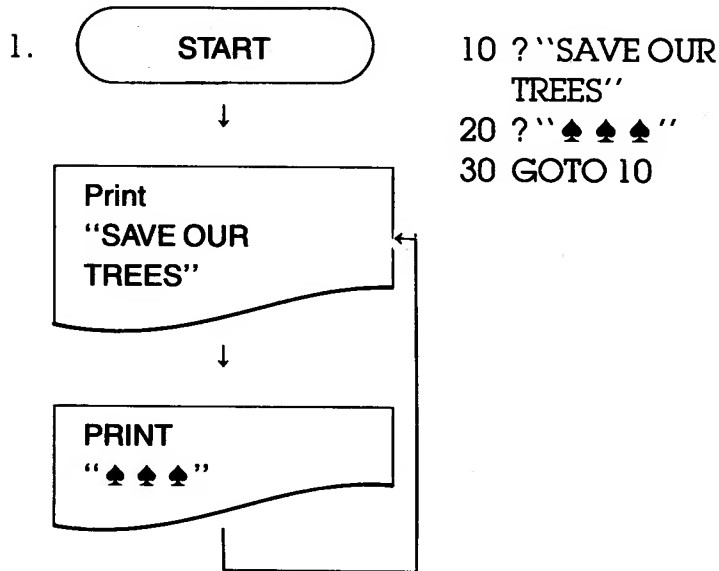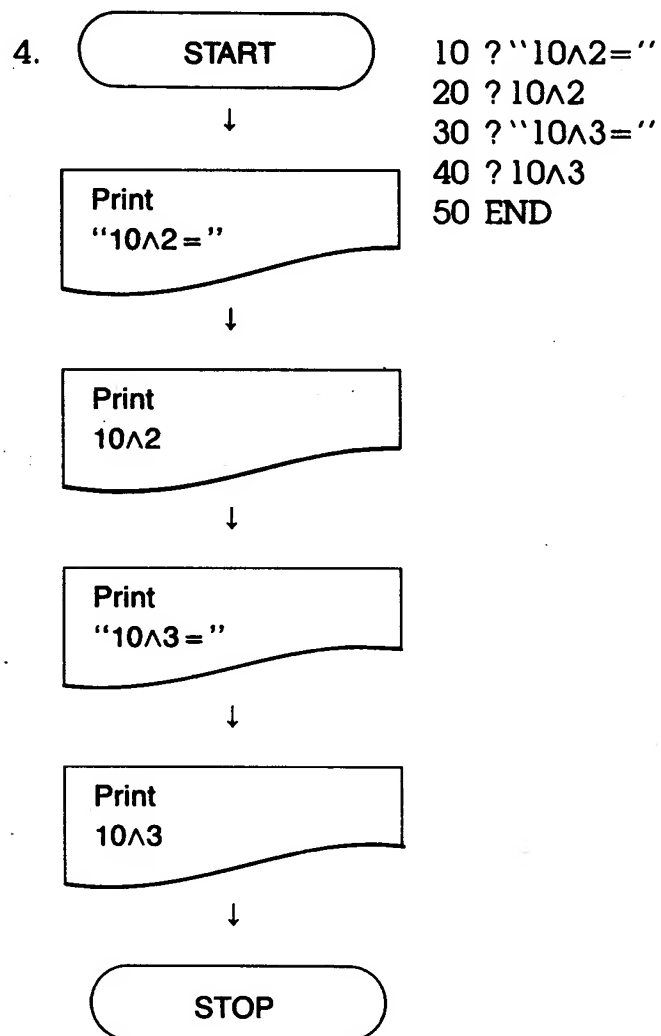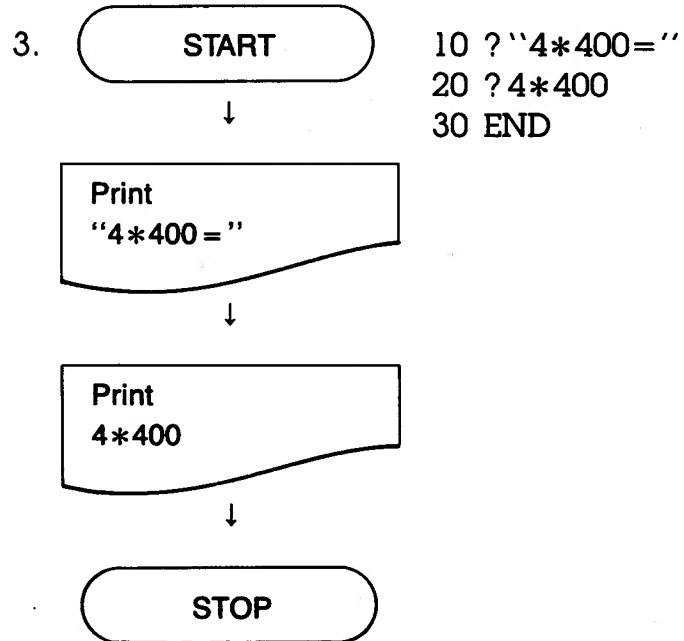
PROGRAMMER'S PASTIME #16

Answers

Write an algorithm in a flow chart for each problem. Then write a BASIC program for ATARI to follow.

1. Tell ATARI to print

PETER PIPER
PICKED A PECK
OF PICKLED PEPPERS

over and over.

Flow chart

```
      START
        ↓
  PRINT "PETER
  PIPER"
        ↓
  PRINT "PICKED
  A PECK"
        ↓
  PRINT "OF
  PICKLED
  PEPPERS"
```
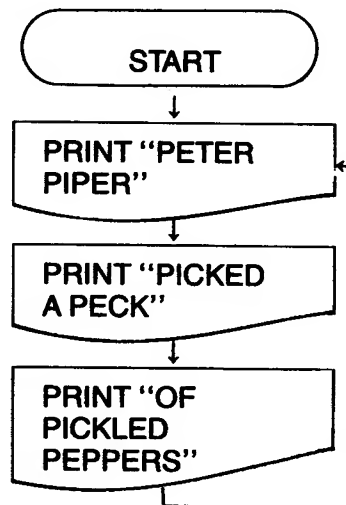
Program

```
10 ? "PETER PIPER"
20 ? "PICKED A PECK"
30 ? "OF PICKLED PEPPERS"
40 GOTO 10
```

2. Tell ATARI to print $100*10-1000=0$

Flow chart

```
      START
        ↓
  PRINT
  "100*10-1000="
        ↓
  PRINT
  100*10-1000
        ↓
      STOP
```
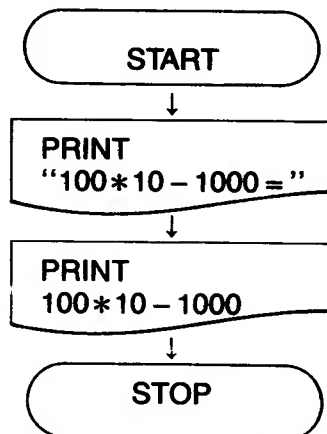
Program

```
10 ? "100*10-1000="
20 ? 100*10-1000
30 END
   OR
20 ? "0"
```

89

3. Tell ATARI to print

> I CAN DO
> 56789*1234
> WHICH EQUALS
> 70077626

Flow chart

```
   ( START )
       ↓
  ┌─────────────┐
  │ PRINT "I    │
  │ CAN DO"     │
  └─────────────┘
       ↓
  ┌─────────────┐
  │ PRINT "56789│
  │ *1234"      │
  └─────────────┘
       ↓
  ┌─────────────┐
  │ PRINT "WHICH│
  │ EQUALS"     │
  └─────────────┘
       ↓
  ┌─────────────┐
  │ PRINT       │
  │ 56789*1234  │
  └─────────────┘
       ↓
   ( STOP )
```
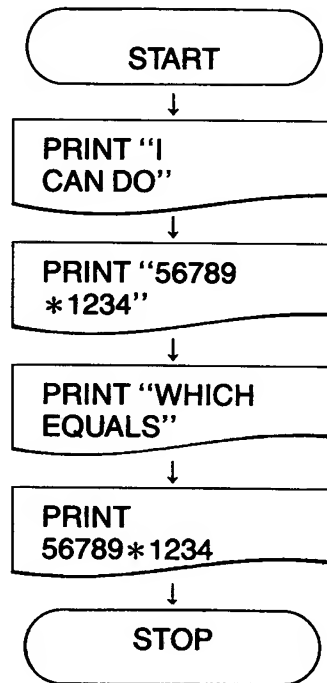
Program

```
10 ?"I CAN DO"
20 ?"56789*1234"
30 ?"WHICH EQUALS"
40 ?56789*1234
50 END
         OR
40 ?"70077626"
```

4. Tell ATARI to print

SOMEWHERE
OVER THE RAINBOW
MANY COMPUTERS TRY
EQUATIONS LIKE
4*10=
(answer to 4*10)

Flow chart

```
   ( START )
       ↓
  PRINT
  "SOMEWHERE"
       ↓
  PRINT "OVER
  THE RAINBOW"
       ↓
  PRINT "MANY
  COMPUTERS
  TRY"
       ↓
  PRINT
  "EQUATIONS
  LIKE"
       ↓
  PRINT
  "4*10="
       ↓
  PRINT
  4*10
       ↓
   ( STOP )
```
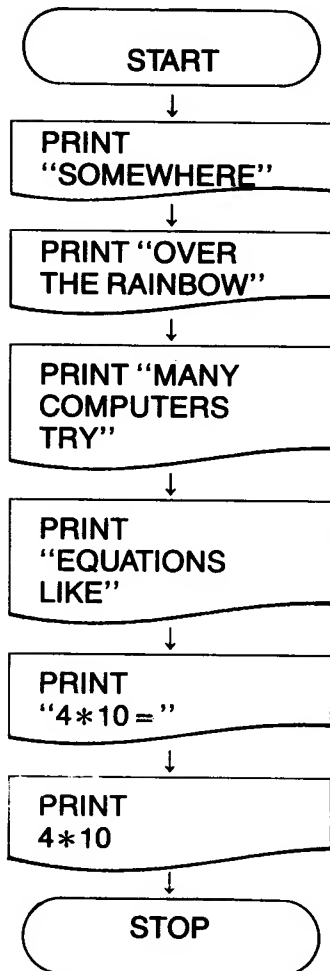
Program

10 ? "SOMEWHERE"
20 ? "OVER THE
    RAINBOW"
30 ? "MANY
    COMPUTERS TRY"
40 ? "EQUATIONS LIKE"
50 ? "4*10="
60 ? 4*10
70 END
         OR
60 ? "40"

Answers

For each program, write what you think ATARI would print.

Example:
```
10 ?"10+500+200="
20 ?10+500+200
30 GOTO 20
```

ATARI would print
```
10+500+200=
710
710
  .  (The dots mean that
  .  710 would be
  .  printed forever.)
```

ATARI would print

1.
```
10 ?"600-400+64="
20 ?600-400+64
30 END
```
```
600-400+64=
264
```

2.
```
10 ?"I LOVE      "
20 ?" YOU !!!      "
30 GOTO 20
```
```
I LOVE
YOU!!!
YOU!!!
YOU!!!
YOU!!!
  .
  .
  .
```

3.
```
10 ?"SHE SELLS"
20 ?"SEA SHELLS"
30 ?"BY THE SEASHORE"
40 ?"I CAN SAY IT ! "
50 GOTO 40
```
```
SHE SELLS
SEA SHELLS
BY THE SEASHORE
I CAN SAY IT!
I CAN SAY IT!
I CAN SAY IT!
  .
  .
  .
```

PROGRAMMER'S PASTIME #18

Answers

For each program, show how ATARI would print the equation on the screen.

Example:

```
10 ? "10+37=",
   10+37
20 END
```

1st Print Zone:

1	2	3	4	5	6	7	8	9	10
1	0	+	3	7	=				

2nd Print Zone:

1	2	3	4	5	6	7	8	9	10
4	7								

1.
```
10 ? "66+33=";
   66+33
20 END
```

1st Print Zone:

1	2	3	4	5	6	7	8	9	10
6	6	+	3	3	=	9	9		

2nd Print Zone:

1	2	3	4	5	6	7	8	9	10

2.
```
10 ? "88-44+2=",
   88-44+2
20 END
```

1st Print Zone:

1	2	3	4	5	6	7	8	9	10
8	8	-	4	4	+	2	=		

2nd Print Zone:

1	2	3	4	5	6	7	8	9	10
4	6								

3.
```
10 ? "100+200+
   300=", 100+
   200+300
20 END
```

1st Print Zone:

1	2	3	4	5	6	7	8	9	10
1	0	0	+	2	0	0	+	3	0

2nd Print Zone:

1	2	3	4	5	6	7	8	9	10
0	=								

The answer, 600, would be printed in columns 1, 2, and 3 in the 3rd print zone.

93

4.
```
10 ?"10*50=";
   10*50
20 ?
30 ?"10*50=",
   10*50
40 END
```

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
1	0	*	5	0	=	5	0	0											
1	0	*	5	0	=					5	0	0							

(1st Print Zone) (2nd Print Zone)

5.
```
10 ?"60/20=",
   60/20
20 ?
30 ?"60/20=";
   60/20
40 END
```

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
6	0	/	2	0	=					3									
6	0	/	2	0	=	3													

(1st Print Zone) (2nd Print Zone)

6.
```
10 ?"5-6=";5-6
20 END
```

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
5	–	6	=	–	1														
5	–	6	=	–	1														

(1st Print Zone) (2nd Print Zone)

7.
```
10 ?"8-7=";8-7
20 ?
30 ?"7-8=";7-8
40 END
```

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
8	–	7	=	1															
7	–	8	=	–	1														

(1st Print Zone) (2nd Print Zone)

Write a program that tells ATARI to print what is seen on the screens below.

Screens   **Programs**

8. ⌜ATARI IS A . . .   NUMBER CRUNCHER   10 ? "ATARI IS A . . .", "NUMBER
   | CRUNCHER"
   | 20 END
   |

9. ⌜10−9=   1   10 ? "10−9=", 10−9
   |9−10=−1   20 ? "9−10="; 9−10
   | 30 END

10. ⌜5−6=−1   10 ? "5−6"; 5−6
    |6−5=   1   20 ? "6−5=", 6−5
    | 30 END

11. ⌜IF YOU CAN DO   8∧8   10 ? "IF YOU CAN DO", "8∧8"
    |YOU'RE A . . .   20 ? "YOU'RE A . . ."
    |WHIZ KID   30 ? "WHIZ KID"
    |WHIZ KID   40 GOTO 30
    |WHIZ KID
    | .
    | .
    | .

COMPONENT 3 FUN PAGE

Crossword Answers:

1 Down: CHART
2 Across: ALGORITHM
3 Down: SINGLE
4 Across: DOUBLE
5 Down: LOOP
6 Across: TEN
7 Down: PROCESS
8 Across: GOTO
9 Down: ZONE
10 Across: DECISION
11 Across: START
12 Across: SEMICOLON
13 Down: STOP
14 Down: COMMA
15 Across: NEW
Down: DECISIONS
Down: STARTING
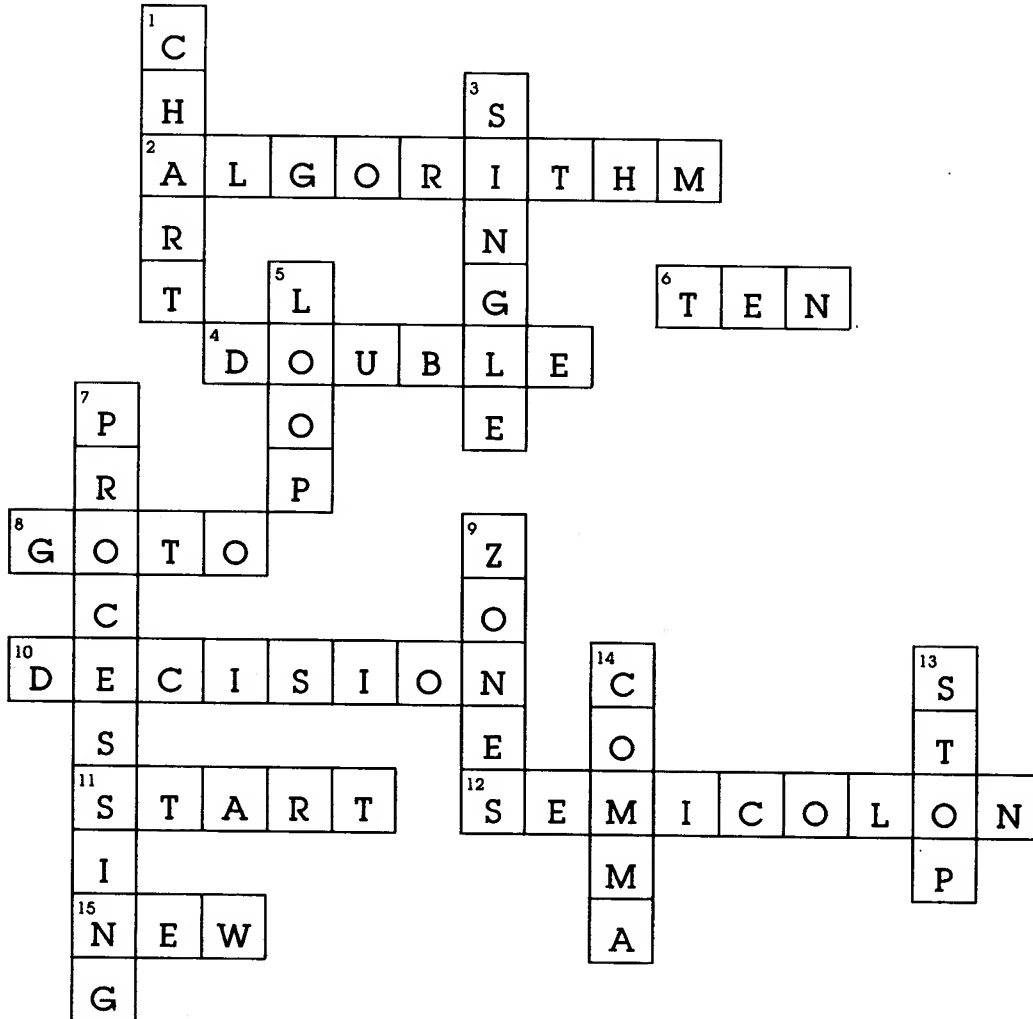
For each LET statement, fill in the CONTENTS of the memory cell mailbox.

1. 10 LET OP=33

   OP
   33

2. 20 LET T2=17

   T2
   17

3. 30 LET M1=3000

   M1
   3000

4. 40 LET D=640

   D
   640

For each LET statement, fill in the ADDRESS and the CONTENTS of the memory cell mailbox.

1. 10 LET FF=99

   FF
   99

2. 20 LET PQ=5

   PQ
   5

3. 30 LET N=0

   N
   0

4. 40 LET W7=62

   W7
   62

Read each variable below. If it follows the rules we learned for writing variables, write "YES." If it does not follow the rules, write "NO."

1. PR	YES	7. X3	YES
2. ITC	YES	8. BB	YES
3. A	YES	9. 2CC	NO
4. E1	YES	10. 23D	NO
5. 3X	NO	11. FDd	NO
6. 7Z	NO	12. KK1	YES

PROGRAMMER'S PASTIME #20

Answers

Read each program. Then write what ATARI would print as the output. If you can, check your answers by running the programs on ATARI.

Program   **Output**

1. 10 LET ZB = 14   14
    20 ? ZB
    30 END

2. 10 LET T2 = 77   T2
    20 ? "T2"
    30 END

3. 10 LET U = 182   U
    20 ? "U"   182
    30 ? U
    40 END

4. 10 LET RC = 7   RC IS   7
    20 ? "RC IS",
    30 ? RC
    40 END

5. 10 LET GG4 = 66   GG4 IS 66
    20 ? "GG4 IS∅";
    30 ? GG4
    40 END

Answers

Read each program. Then write what ATARI would print as the output. If you can, check your answers by running the programs on ATARI. Pay close attention to safe and unsafe variables!

Program

1.
```
10  LET RB4=40
20  LET RB5=50
30  LET RB1=10
40  ? RB5; "ɸISɸ";
50  ? RB1; "ɸMORE";
60  ? "THANɸ"; RB4
70  END
```

2.
```
10  LET T=5
20  LET V=25
30  ? "THE SQUARE
       ROOT OFɸ";
40  ? V; "ɸISɸ"; T
50  END
```

3.
```
10  ? "MY FAVORITE
       NUMBER ISɸ";
20  LET D=333
30  ? D
40  GOTO 30
```

4.
```
10  ? "MY FAVORITE
       NUMBER ISɸ";
20  LET D=333
30  ? D;
40  GOTO 30
```

Output

1. 50 IS 10 MORETHAN 40

(Notice that no blank space —ɸ—was left after "MORE" or before "THAN". When this happens the two words run together as "MORETHAN" in the output.)

2. THE SQUARE ROOT OF 25 IS 5

3.
```
MY FAVORITE
NUMBER IS 333
333
333
333
  ⋮
```

4.
```
MY FAVORITE
NUMBER IS 333333
33333333333333333
333333333333333
333333333333333
  ⋮    ⋮    ⋮
```

Answers

Read each program. Then write what ATARI would print as the output.

Program	**Output**
1. 10 LET N=12	2
20 LET R=6	
30 ? N/R	
40 END	
2. 10 LET N=12	N+R=18
20 LET R=6	
30 ? ''N+R='';N+R	
40 END	
3. 10 LET N=12	12+6=18
20 LET R=6	
30 ? N;''+'';R;''=''; N+R	
40 END	
4. 10 LET Q=10	2
20 LET R=20	10+30=40
30 LET S=30	20=30-10
40 LET T=40	
50 ? T/R	
60 ? Q;''+'';S;''='';T	
70 ? S-Q;''='';S; ''-'';Q	
80 END	

PROGRAMMER'S PASTIME #23

For each LET statement, fill in the contents of the memory cell mailboxes.

1. 10 LET P=41
   20 LET Q=P

   P
   41

   Q
   41

2. 10 LET A=36
   20 LET E=16
   30 LET I=A

   A
   36

   E
   16

   I
   36

3. 10 LET L=4
   20 LET M=2
   30 LET N=4+2

   L
   4

   M
   2

   N
   6

4. 10 LET B1=3
   20 LET B2=6
   30 LET B3=B1+B2

   B1
   3

   B2
   6

   B3
   9

5. 10 LET AB=10
   20 LET AC=15
   30 LET AD=AC+5
   40 LET AE=AB+10

   AB
   10

   AC
   15

   AD
   20

   AE
   20

6. 10 LET GP=19
   20 LET GQ=GP−4
   30 LET GR=GQ+4
   40 LET GS=GR∗1

   GP
   19

   GQ
   15

   GR
   19

   GS
   19

Answers

Read each program. Then write what ATARI would print as the output. Check your answers by running the programs on ATARI.

Program	Output	
1. 10 LET PJ=17	51	
20 LET J2=34		
30 LET J4=PJ+J2		
40 ? J4		
50 END		
2. 10 LET B=2	2	
20 ? B	100	
30 LET B=100		
40 ? B		
50 END		
3. 10 LET X1=2	5	
20 LET X2=X1*5		
30 LET X3=X2/X1		
40 ? X3		
50 END		
4. 10 LET E6=3	PRODUCT	QUOTIENT
20 LET E7=12	36	4
30 ?"PRODUCT", "QUOTIENT"		
40 ?E6*E7, E7/E6		
50 END		

Program	Output
5. 10 LET HI=16	30
20 LET HJ=HI+4	
30 ?HJ+10	
40 END	
6. 10 LET M=16	30
20 LET N=14	28
30 ?M+N	
40 LET N=12	
50 ?M+N	
60 END	
7. 10 LET Z1=8	10
20 LET Z2=Z1−2	
30 ?Z2+Z1/2	
40 END	
8. 10 LET T1=6	7
20 LET T1=7	7
30 ?T1	7
40 GOTO 30	⋮
50 END	
9. 10 LET J=11	39
20 LET K=22	
30 LET J=17	
40 ?K+J	
50 END	

PROGRAMMER'S PASTIME #25

Answers

Read each program. Then write what ATARI would print as the output. Check your answers by running the programs on ATARI.

Program

1. 10 LET A=100
    20 LET B=A/25
    30 LET C=B*A
    40 ? "B="; B
    50 ? "C="; C
    60 ? "IF A+B+C=";
       A+B+C
    70 ? "THEN A="; A
    80 END

2. 10 LET Q1=8
    20 LET QZ=4
    30 ? Q1/QZ
    40 LET Q1=12
    50 ? Q1/QZ
    60 END

3. 10 LET C=9
    20 LET D=8
    30 LET C=7
    40 ? C
    50 ? D
    60 END

Output

B=4
C=400
IF A+B+C=504
THEN A=100

2
3

7
8

Answers

In each program there are one or more mistakes. Find the mistake(s) and circle the line number where you found it. Then write the statement the correct way in the space to the right.

Program	**Correction**

Example:

```
⑩ LET 4=D          10 LET D=4
20 ? D
30 END
```

1.
```
   10 LET L=44
   ㉚ LET 6=M         20 LET M=6
   ㉚ ?L+D            30 ?L+M
   40 END
```

2.
```
   ⑩ ?Y              10 LET Y=66
   ㉚ LET Y=66        20 ?Y
   30 END
```

3.
```
   ⑩ LET 5B=6        10 LET B5=6
   20 LET H=3
   20 ?H
   40 END
```

4.
```
   ⑩ ?"A+B"; A+B      10 LET A=3
   ㉚ LET A=3          20 LET B=4
   ㉚ LET B=4          30 ?"A+B";A+B
   40 END
```

Program	Correction
5. 10 LET XY=5	
20 LET 3=VW	20 LET VW=3
30 ?XY+VW	
40 END	
6. 10 LET X=99	
20 LET C=4	
?X−C	30 ?X−C
40 END	
7. 10 LET D+2=10	10 LET D=2+10
20 LET E=4	
30 ?D−E	
40 END	
8. 10 LET 3Y=2	10 LET Y3=2
20 LET B1=9	
30 ?B1	
40 END	

PROGRAMMER'S PASTIME #27

Using any shortcuts you have learned so far, rewrite the long programs below to make them as short as possible.

Program   **Shortcut**

Example:
```
10 LET B=49          10 LET B=49: LET E=409
20 LET E=409         20 ?B+E
30 PRINT B+E         30 END
40 END
```

```
1. 10 LET R=50        10 LET R=50:LET N=22
   20 LET N=22        20 ?R−N, R+N
   30 ?R−N            30 END
   40 ?R+N
   50 END
```

```
2. 10 LET S1=98       10 LET S1=98:LET S2=89:
   20 LET S2=89          LET S3=889
   30 LET S3=889      20 ?"S3−S2−S1=";S3−S2−S1,
   40 PRINT "S3−S2−S1=";   "S2*S1=";S2*S1
      S3−S2−S1        30 END
   50 PRINT "S2*S1=";
      S2*S1
   60 END
```

```
3. 10 LET U=40        10 LET U=40:LET T=20
   20 LET T=20        20 ?U+T, U−T, U*T, U/T
   30 ?U+T            30 END
   40 ?U−T
   50 ?U*T
   60 ?U/T
   70 END
```

Program	Shortcut
4. 10 LET Y2=2	10 LET Y2=2:LET Y3= 22:
20 LET Y3=22	LET Y4=Y2+2: LET
30 LET Y4=Y2+2	Y5=Y3+22
40 LET Y5=Y3+22	20 ?"Y4="; Y4,
50 PRINT "Y4=";	"Y5="; Y5
Y4	30 END
60 PRINT "Y5=";	
Y5	
70 END	
5. 10 LET D2=9	10 LET D2=9:
20 LET D4=18	LET D4=18
30 ?"D2+D4=";	20 ?"D2+D4=";
40 ?D2+D4	D2+D4, "D4- D2=";
50 ?"D4-D2=";	D4-D2
60 ?D4-D2	30 END
70 END	

PROGRAMMER'S PASTIME #28

For each E notation number, write the whole number or decimal that it stands for.

E Notation   **Whole number or decimal**

Example: 7.982 E+07   79820000

	E Notation	Whole number or decimal
1.	3.26 E+11	326000000000
2.	1.23 E−04	.000123
3.	7.2 E+08	720000000
4.	4.679 E−06	.000004679
5.	2.22 E+07	22200000
6.	6.46982 E−03	.00646982
7.	5.3211 E+12	5321100000000
8.	9.011 E−05	.00009011
9.	8.0045 E+13	80045000000000
10.	1.467 E−07	.0000001467
11.	8.006 E+09	8006000000
12.	6.002 E−03	.006002
13.	3.9826 E−05	.000039826
14.	1.976345 E+08	197634500
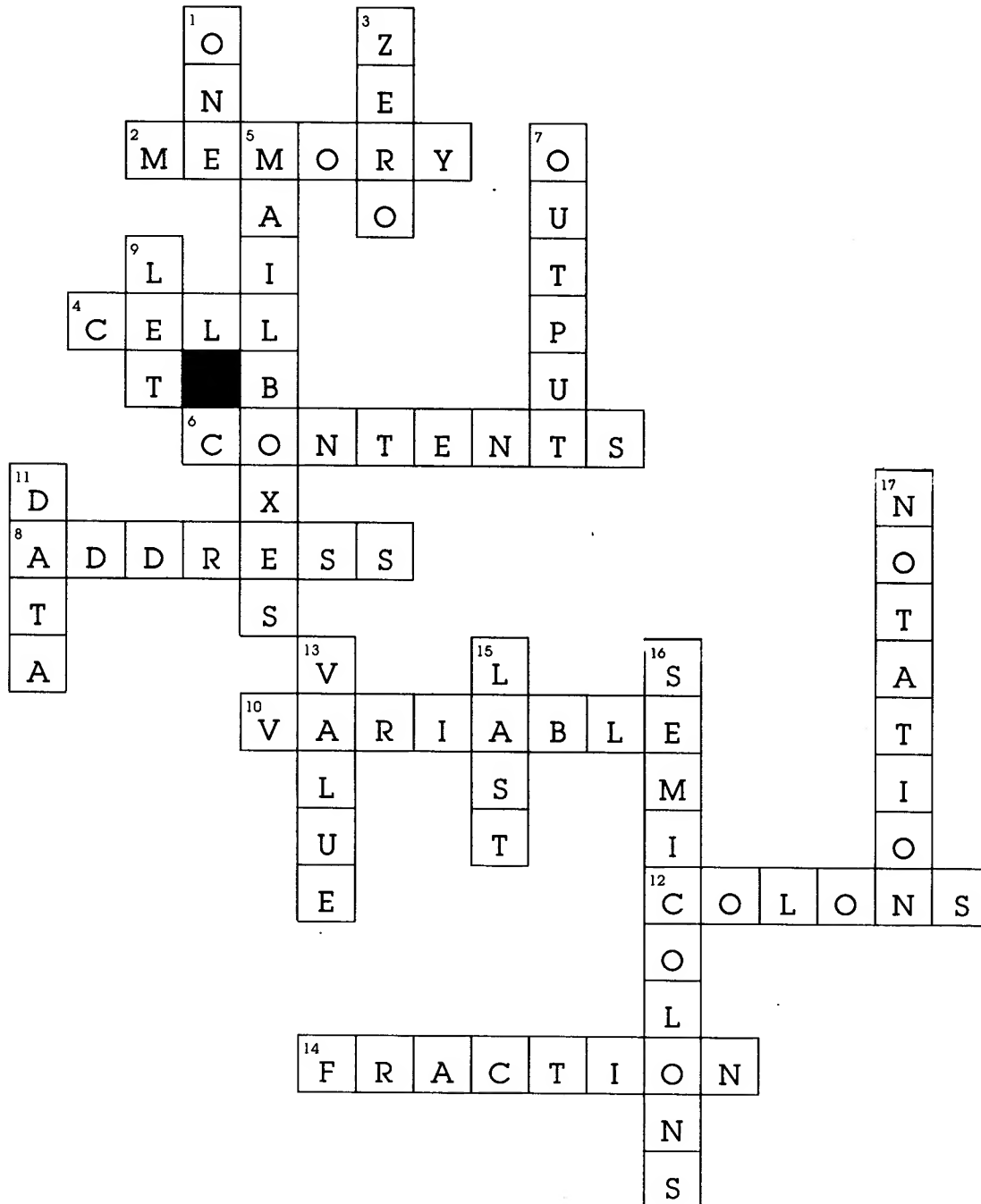
Answers

A crossword puzzle grid with the following answers:

- 1 Down: ON
- 3 Down: ZERO
- 2 Across: MEMORY
- 5 Down: MAILBOX
- 7 Down: OUTPUT
- 9 Down: LET
- 4 Across: CELL
- 6 Across: CONTENTS
- 11 Down: DATA
- 8 Across: ADDRESS
- 17 Down: NOTATION
- 13 Down: VALUE
- 15 Down: LAST
- 16 Down: SEMICOLONS
- 10 Across: VARIABLE
- 12 Across: COLONS
- 14 Across: FRACTION

Answers

Read each program. Then write what you think ATRARI would print as the OUTPUT. Run the programs on ATARI to check your answers.

Program   **Output**

1. 10 FOR Q=2 TO 6   2
    20 ? Q   3
    30 NEXT Q   4
    40 END   5
                                     6

2. 10 FOR Q=2 TO 4   Q=2
    20 ? "Q=";Q   Q=3
    30 NEXT Q   Q=4
    40 END

3. 10 FOR A=1 TO 5   HELLO FRIEND
    20 ? "HELLO FRIEND!"   HOW ARE YOU?
    30 ? "HOW ARE YOU?"   HELLO FRIEND
    40 NEXT A   ⋮
    50 END   (3 more times)

4. 10 FOR D=1 TO 3   1
    20 ? D   11
    30 ? D+10   2
    40 NEXT D   12
    50 END   3
                                     13

5. 10 LET P=3   3+1=4
    20 FOR Q=1 TO 3   3+2=5
    30 ? P;"+";Q;"=";   3+3=6
        P+Q
    40 NEXT Q
    50 END

Program	Output		
6.10 FOR B=1 TO 5	B	B+B	B*B
20 ?"B", "B+B", "B*B"	1	2	1
30 ?B, B+B, B*B	B	B+B	B*B
40 NEXT B	2	4	4
50 END	B	B+B	B*B
	3	6	9

(and so on . . . up to 5)

Program	Output
7.10 ?"MULTIPLICATION TABLE FOR 7"	MULTIPLICATION TABLE FOR 7
20 FOR K=1 TO 12	1 TIMES 7=7
30 ?K; "TIMES 7="; K*7	2 TIMES 7=14
40 NEXT K	3 TIMES 7=21
50 END	4 TIMES 7=28

(and so on . . . up to 12 TIMES 7)

Program	Output
8.10 FOR G=1 TO 10	♡
20 ?"♡"	♡
30 NEXT G	♡
40 END	♡
	♡
	⋮

(up to 10)

Program	Output
9.10 FOR G=1 TO 10	♡ ♡ ♡ ♡ ♡ ♡ ♡ ♡ ♡ ♡
20 ?"♡";	
30 NEXT G	
40 END	

Program	Output	
10.10 FOR S=1 TO 10	1	1
20 LET S=S*S	4	1
30 ?S, S/S	25	1
40 NEXT S		
50 END		

Can you explain how this program works?

LINE 20 increases the value of S so that by the third time through the loop S *S=25. (In the second loop S was changed to 4(2*2) and therefore goes to 5 next.) The program stops because 25>10.

PROGRAMMER'S PASTIME #30

Write a program for each flow chart. Be sure to use a FOR–NEXT loop. Run your programs on ATARI to make sure they work.

Flow chart   **Program**

1.   (START)

        ↓

     ┌──────────┐
     │ FOR X = │ ←
     │ 1 TO 10 │
     └──────────┘

        ↓

     ┌──────────────┐
     │ PRINT   │
     │ "I'M A   │
     │ PROGRAMMER" │
     └──────────────┘

        ↓

     ┌──────────┐
     │ NEXT X   │
     └──────────┘

        ↓
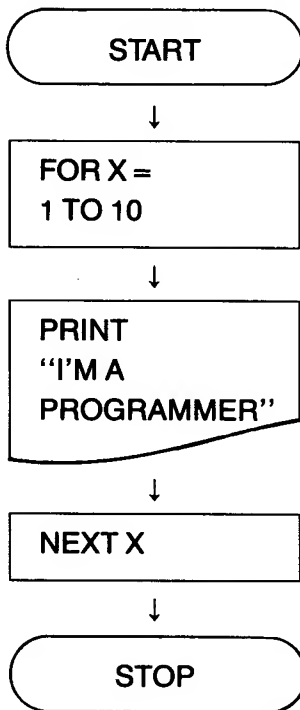
     (STOP)

10 FOR X = 1 TO 10
20 ? "I'M A
      PROGRAMMER"
30 NEXT X
40 END

2.   (START)

        ↓

     ┌──────────┐
     │ FOR Y = │ ←
     │ 1 TO 50 │
     └──────────┘

        ↓

     ┌──────────┐
     │ PRINT   │
     │ Y   │
     └──────────┘

        ↓

     ┌──────────┐
     │ NEXT Y   │
     └──────────┘

        ↓
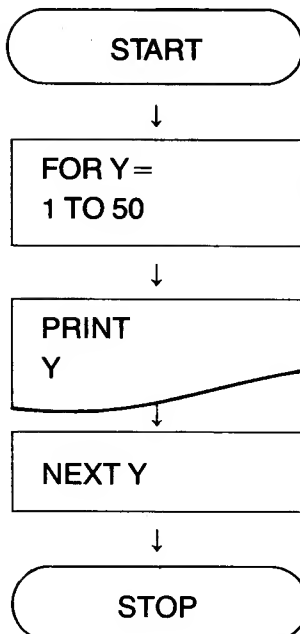
     (STOP)

10 FOR Y = 1 TO 50
20 ? Y
30 NEXT Y
40 END

113

Flow chart	Program

3.

```
  START
    ↓
  PRINT
  "F", "F−F",
  "F/F"
    ↓
  FOR F =
  1 TO 10        ←┐
    ↓             │
  PRINT           │
  F, F−F, F/F     │
    ↓             │
  NEXT F        ──┘
    ↓
  STOP
```
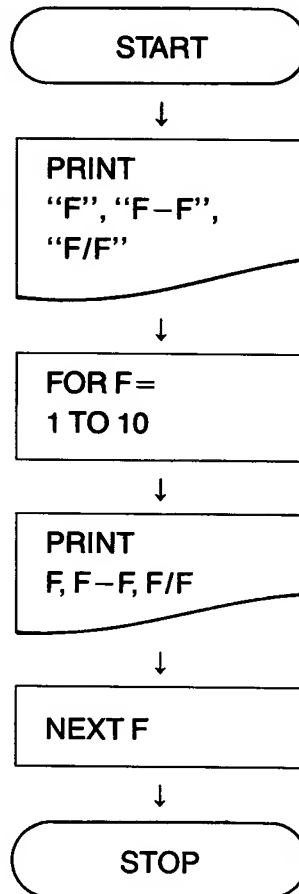
```
10 ? "F", "F−F", "F/F"
20 FOR F = 1 TO 10
30 ? F, F−F, F/F
40 NEXT F
50 END
```

4.

```
START
  ↓
PRINT
"MULTIPLICATION
TABLE FOR 8"
  ↓
FOR J=
1 TO 12
  ↓
PRINT
J*8
  ↓
NEXT J
  ↓
STOP
```

10 ? "MULTIPLICATION
     TABLE FOR 8"
20 FOR J=1 TO 12
30 ? J*8
40 NEXT J
50 END

115

For each program description, write an algorithm in flow chart form. Then write the program. Run each program on ATARI to make sure it works.

Description	Flow chart	Program
5. Add something new to the program in #4 so it prints: J; "TIMES 8="; J*8 each time the loop is done.		10 ? "MULTIPLICATION TABLE FOR 8" 20 FOR J=1 TO 12 30 ? J; "⌀TIMES 8=⌀"; J*8 40 NEXT J 50 END
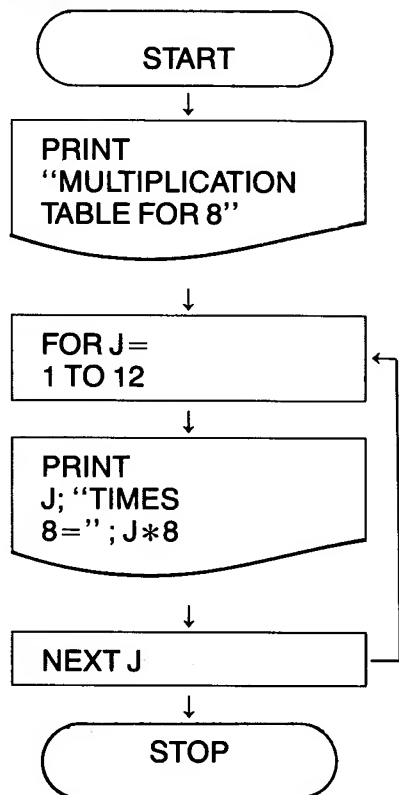
Flow chart #5:
- START
- PRINT "MULTIPLICATION TABLE FOR 8"
- FOR J= 1 TO 12
- PRINT J; "TIMES 8="; J*8
- NEXT J
- STOP

| 6. Write a program that prints the numbers from 1 to 20, their squares (X*X), and their cubes (X*X*X). | | 10 FOR X=1 TO 20
20 ? X, X*X, X*X*X
30 NEXT X
40 END |
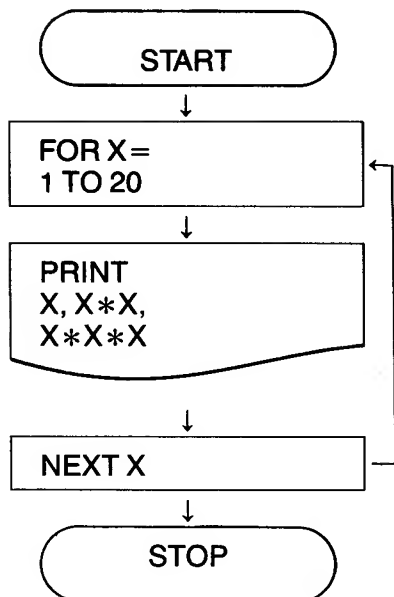
Flow chart #6:
- START
- FOR X= 1 TO 20
- PRINT X, X*X, X*X*X
- NEXT X
- STOP

Description	Flow Chart	Program
7. Write a program that prints * 20 times on one line.		10 FOR N = 1 TO 20 20 ?"*"; 30 NEXT N 40 END
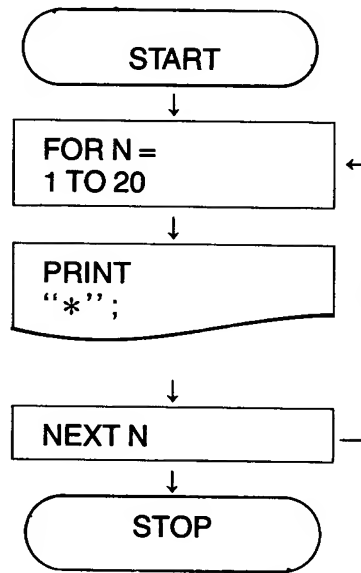
```
START
  ↓
FOR N =
1 TO 20          ←
  ↓
PRINT
"*";
  ↓
NEXT N
  ↓
STOP
```

Description	Flow Chart	Program
8. Write a program that introduces D=5 and P=1 to 5. Make the program add D plus each value of P, and print the sums of D+P.		10 LET D=5 20 FOR P=1 TO 5 30 ?D+P 40 NEXT P 50 END
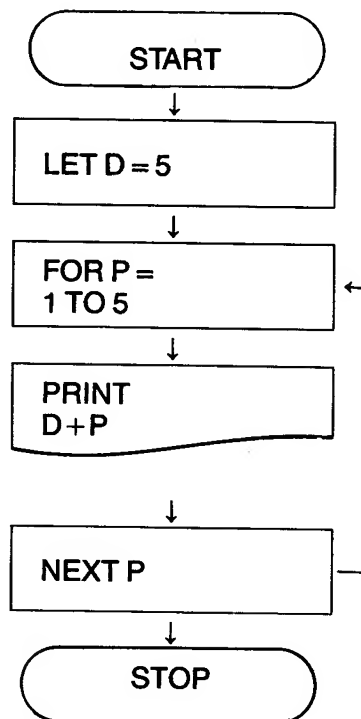
```
START
  ↓
LET D = 5
  ↓
FOR P =
1 TO 5           ←
  ↓
PRINT
D+P
  ↓
NEXT P
  ↓
STOP
```

PROGRAMMER'S PASTIME #31

Answers

In each program there is one mistake. Find the mistake and circle the line number where you found the mistake. Then write the statement the correct way in the space to the right.

Program

1. ⑩ FOR P=1 – 40
   20 ?P
   30 NEXT P
   40 END

2. ⑩ FOR W IS 6 TO 30
   20 ?W
   30 NEXT W
   40 END

3. 10 FOR E=3 TO 10
   20 ?E
   ㉚ E NEXT
   40 END

4. 10 FOR L=1 TO 4
   20 ?L
   30 ?L*2
   ㊵
   50 END

5. 10 LET G=4
   20 FOR H=5 TO 9
   30 ?G+H
   ㊵ NEXT G
   50 END

Correction

10 FOR P=1 TO 40

10 FOR W=6 TO 30

30 NEXT E

40 NEXT L

40 NEXT H

118

Answers

Read each program. Write what you think ATARI
would print as the output. Run the programs on
ATARI to check your answers.

PROGRAM	OUTPUT
1. 10 FOR F=0 TO 8	0
STEP 2	2
20 ? F	4
30 NEXT F	6
40 END	8
2. 10 FOR J=18 TO 0	18
STEP−3	15
20 ? J	12
30 NEXT J	9
40 END	6
	3
	0
3. 10 FOR B2=3 TO 21	HOWDY
STEP 3	HOWDY
20 ? "HOWDY"	HOWDY
30 NEXT B2	HOWDY
40 END	HOWDY
	HOWDY
	HOWDY
4. 10 LET N=5	1
20 FOR T=1 TO N	2
30 ? T	3
40 NEXT T	4
50 END	5
5. 10 LET M2=10	0
20 FOR S=0 TO 12	2
STEP M2/5	4
30 ? S	6
40 NEXT S	8
50 END	10
	12

Program	Output
Program	**Output**
6. 10 ? "IF SEPT 1 IS A SUNDAY THEN"	IF SEPTEMBER 1 IS A SUNDAY THEN
20 FOR JJ=1 TO 31 STEP 7	SEPT 1 IS A SUNDAY
30 ? "SEPT∅"; JJ; "∅IS A SUNDAY"	SEPT 8 IS A SUNDAY
40 NEXT JJ	SEPT 15 IS A SUNDAY
50 END	SEPT 22 IS A SUNDAY
	SEPT 29 IS A SUNDAY
7. 10 LET PX=8	0
20 FOR A7=0 TO 10 STEP PX/4	2
30 ? A7	4
40 NEXT A7	6
50 END	8
	10
8. 10 FOR ZZ=1 TO 14 STEP 4	1
20 ? ZZ	5
30 NEXT ZZ	9
40 END	13
9. 10 FOR BD=20 TO 2 STEP −5	20
20 ? BD	15
30 NEXT BD	10
40 END	5

PROGRAMMER'S PASTIME #33

For each program description, write an algorithm in flow chart form. Then write the program. Run each program on ATARI to make sure it works.

Description	**Flow Chart**	**Program**

1. Write a program that tells ATARI to count from 0 to 40 by fours.

```
( START )
   ↓
FOR X =
0 TO 40 STEP 4
   ↓
PRINT X
   ↓
NEXT X
   ↓
( STOP )
```
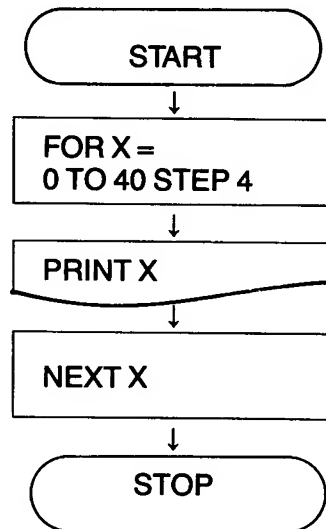
```
10  FOR X = 0 TO 40
    STEP 4
20  ? X
30  NEXT X
40  END
```

2. Write a program that tells ATARI to count backwards from 8 to 0.

```
( START )
   ↓
FOR X =
8 TO 0 STEP – 1
   ↓
PRINT X
   ↓
NEXT X
   ↓
( STOP )
```
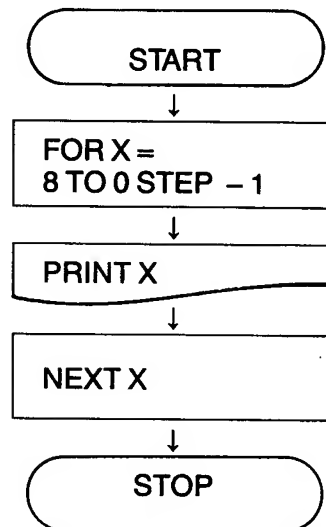
```
10  FOR X = 8 TO 0
    STEP – 1
20  ? X
30  NEXT X
40  END
```

Description	Flow Chart	Program

3. Write a program that tells ATARI to print "HELLO" five times. Use a STEP statement.

```
START
  ↓
FOR X =
0 TO 25 STEP 5
  ↓
PRINT "HELLO"

  ↓
NEXT X
  ↓
STOP
```
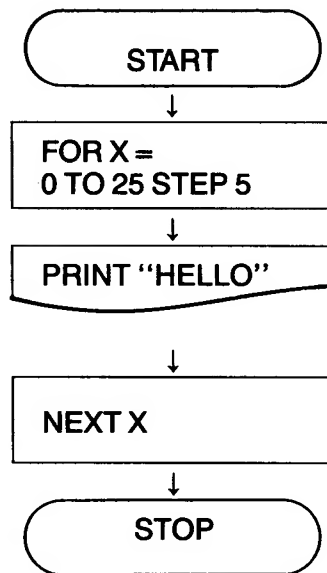
EXAMPLE

```
10 FOR X=0 TO 25
   STEP 5
20 ? "HELLO"
30 NEXT X
40 END
```

4. Write a program that tells the ATARI to print the numbers 0 through 21 with STEP N/4. Make N=12.

```
START
  ↓
LET N=12
  ↓
FOR X=0 TO 21
STEP N/4
  ↓
PRINT X
  ↓
NEXT X
  ↓
STOP
```
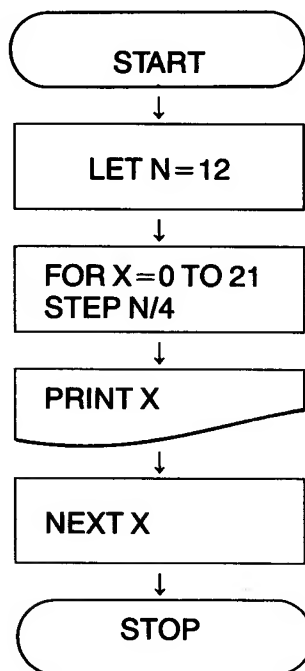
```
10 LET N=12
20 FOR X=0 TO 21
   STEP N/4
30 ? X
40 NEXT X
50 END
```

5. Write a program that tells the ATARI to print the numbers 1, 4, 7, 10 and 13. Use a STEP statement.

```
START
  ↓
FOR X=1 TO 13
STEP 3
  ↓
PRINT X
  ↓
NEXT X
  ↓
STOP
```
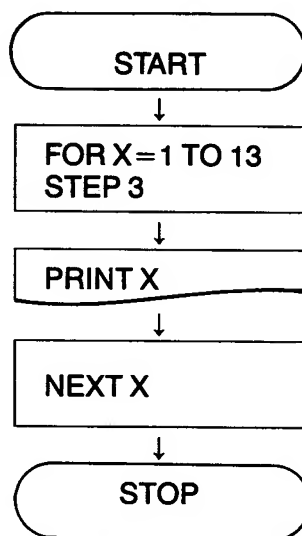
```
10 FOR X=1 TO 13
   STEP 3
20 ?X
30 NEXT X
40 END
```

Answers

Study each program. Write what you think ATARI would print as the OUTPUT. Run each program to check your answers.

Program	Output
1. 10 LET C=0	THINK
20 FOR FL=1 TO 100	1
STEP 10	THINK
30 ? "THINK"	2
40 LET C=C+1	THINK
50 ? C	3
60 NEXT FL	THINK
70 END	4

$$\vdots$$

(and so on...to $\dfrac{\text{THINK}}{10}$)

Program	Output
2. 10 LET C=0	BRAIN POWER
20 ? "BRAIN POWER"	1
30 LET C=C+1	BRAIN POWER
40 ? C	2
50 GOTO 20	BRAIN POWER
	3

$$\vdots$$

Program	Output
3. 10 LET C=0	1
20 FOR FL=1 TO 8	AWESOME
30 LET C=C+1	2
40 ? C	AWESOME
50 ? "AWESOME"	3
60 NEXT FL	AWESOME
70 END	$\vdots$
	8
	AWESOME

Program	Output
4. 10 LET C=0	RAINBOW
20 FOR Z=1 TO 50 STEP 10	RAINBOW
30 ?"RAINBOW"	RAINBOW
40 LET C=C+1	RAINBOW
50 NEXT Z	RAINBOW
60 ?"I PRINTED"	I PRINTED
70 ?"RAINBOW"	RAINBOW
80 ?C;"TIMES"	5 TIMES
90 END	
5. 10 LET C=0	1
20 FOR FL=1 TO 4	JELLY BEANS
30 LET C=C+1	2
40 ?C	JELLY BEANS
50 ?"JELLY BEANS"	3
60 NEXT FL	JELLY BEANS
70 ?"A TOTAL OFø"; C;"øJELLY	4
BEANS"	JELLY BEANS
80 END	A TOTAL OF 4 JELLY BEANS

Answers

Write a program for each flow chart. Then run the programs.

Flow Chart   **Program**

1.
```
   ( START )
       ↓
  CLEAR SCREEN
       ↓
  PRINT "THIS IS
  FUN!"
       ↓
  FOR–NEXT
  TIME LOOP
       ↓
  CLEAR SCREEN
       ↓
   ( STOP )
```
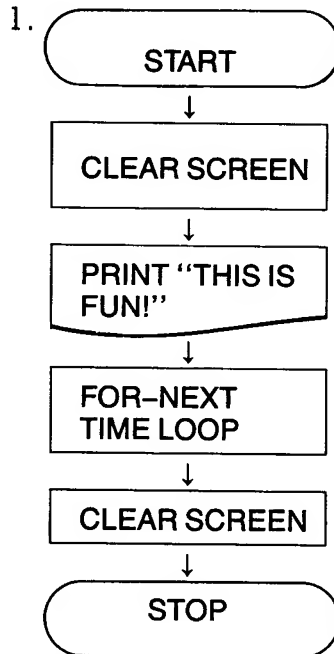
```
10 ?" [ESC] [SHIFT] [CLEAR <] "
20 ?"THIS IS FUN!"
30 FOR T=1 TO
   1000:NEXT T
40 ?" [ESC] [SHIFT] [CLEAR <] "
50 END
```

2.
```
   ( START )
       ↓
  CLEAR SCREEN
       ↓
  FOR N = 1 TO 10  ←
       ↓
  PRINT N
       ↓
  FOR–NEXT
  TIME LOOP
       ↓
  CLEAR SCREEN
       ↓
  NEXT N
       ↓
   ( STOP )
```
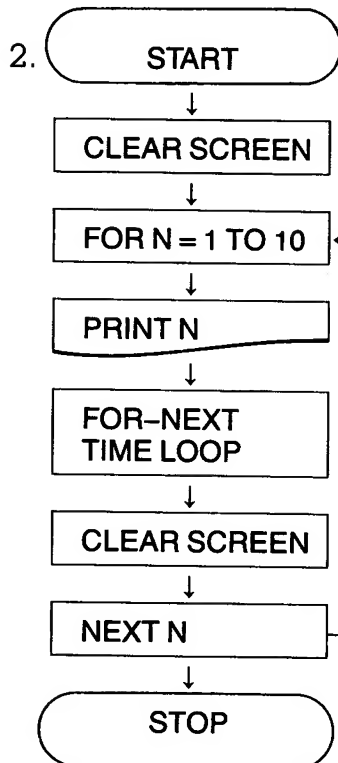
```
10 ?" [ESC] [SHIFT] [CLEAR <] "
20 FOR N=1 TO 10
30 ?N
40 FOR T=1 TO 1000:
   NEXT T
50 ?" [ESC] [SHIFT] [CLEAR <] "
60 NEXT N
70 END
```

Flow Chart

Program

3.
```
START
  ↓
CLEAR SCREEN
  ↓
PRINT
"LOVE IS"
  ↓
FOR-NEXT     →   CLEAR      →   PRINT
TIME LOOP        SCREEN         "♡ ♡ ♡"  ←┐
                                         └┘
```

```
10 ?" [ESC] [SHIFT] [CLEAR<] "
20 ?"LOVE IS"
30 FOR T= 1 TO 1000:
   NEXT T
40 ?" [ESC] [SHIFT] [CLEAR<] "
50 ?"♡ ♡ ♡"
60 GOTO 50
```

4.
```
START
  ↓
CLEAR SCREEN
  ↓
FOR N = 1 TO 100 ←┐
  ↓               │
PRINT             │
"COUNTING"        │
  ↓               │
PRINT N           │
  ↓               │
FOR–NEXT          │
TIME LOOP         │
  ↓               │
CLEAR SCREEN      │
  ↓               │
NEXT N  ──────────┘
  ↓
CLEAR SCREEN
  ↓
STOP
```

```
10 ?" [ESC] [SHIFT] [CLEAR<] "
20 FOR N= 1 TO 100
30 ?"COUNTING"
40 ?N
50 FOR T= 1 TO 1000:
   NEXT T
60 ?" [ESC] [SHIFT] [CLEAR<] "
70 NEXT N
80 ?" [ESC] [SHIFT] [CLEAR<] "
90 END
```

Flow Chart

5.

```
      ( START )
          ↓
   ┌──────────────┐
   │ CLEAR SCREEN │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ FOR N = 0 TO 50 │
   │ STEP 5       │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ PRINT "COUNT │
   │ BY FIVES"    │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ SKIP A LINE  │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ PRINT N      │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ FOR–NEXT     │
   │ TIME LOOP    │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ CLEAR SCREEN │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ NEXT N       │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ CLEAR SCREEN │
   └──────────────┘
          ↓
      ( STOP )
```

Program

```
10  ? " [ESC] [SHIFT] [CLEAR <] ''
20  FOR N = 0 TO 50
    STEP 5
30  ? "COUNT BY
    FIVES"
40  ?
50  ? N
60  FOR T = 1 TO 1000:
    NEXT T
70  ? " [ESC] [SHIFT] [CLEAR <] ''
80  NEXT N
90  ? " [ESC] [SHIFT] [CLEAR <] ''
100 END
```

You have learned how to program ATARI to move down a number of lines on the screen and then begin printing. To do this, you used a statement like this:

20 ?:?:?:?

This tells ATARI to move down 4 lines.

Now using the tricks you have learned to make ATARI clear the screen, and move the writing down several lines, write a program for the following flow charts. Try the programs out on ATARI.

Flow Chart

1.

START

↓

CLEAR SCREEN

↓

MOVE 10
LINES DOWN

↓

PRINT
"SMART
STUFF"

↓

FOR T=1 TO
1000:NEXT T

↓

CLEAR SCREEN

↓

STOP
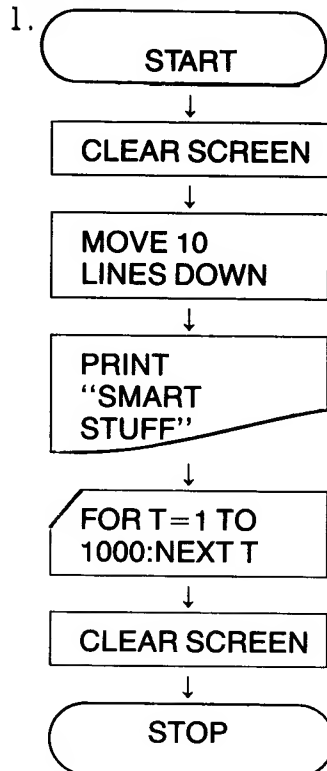
Program

10 ? " [ESC] [SHIFT] [CLEAR <] "
20 ?:?:?:?:?:?:?:?:?:?
30 ? "SMART STUFF"
40 FOR T=1 TO 1000:
    NEXT T
50 ? " [ESC] [SHIFT] [CLEAR <] "
60 END

Flow Chart

2.

```
  ( START )
      ↓
  CLEAR SCREEN
      ↓
  PRINT "WHAT A
  GREAT TRICK!"
      ↓
  MOVE DOWN 4
  LINES
      ↓
  FOR T = 1 TO
  1000:NEXT T
```
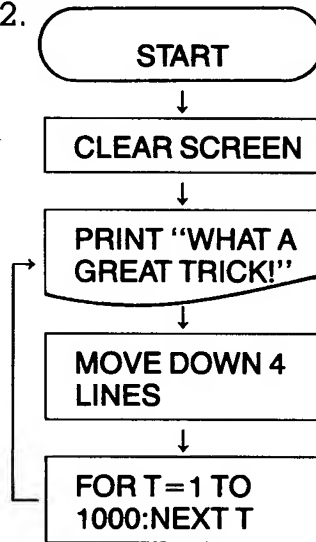
Program

10 ? " [ESC] [SHIFT] [CLEAR <] "
20 ? "WHAT A GREAT
     TRICK!"
30 ?:?:?:?
40 FOR T = 1 TO 1000:
     NEXT T
50 GOTO 20

Write a program for each flow chart, then run
your programs on ATARI to make sure they work.

Flow Chart   **Program**

1.

```
     ( START )
         ↓
  [ CLEAR SCREEN ] ←┐
         ↓          │
  [ FOR–NEXT        │
    TIME LOOP ]     │
         ↓          │
  [ PRINT "OFF      │
    AND ON" ] ──────┘
         ↓
```
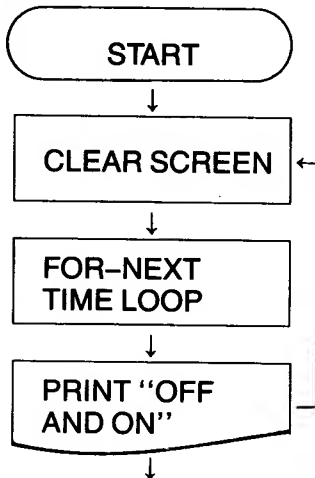
10 ? " [ESC] [SHIFT] [CLEAR <] "
20 FOR T = 1 TO 100:
   NEXT T
30 ? "OFF AND ON"
40 GOTO 10

This is the basic algorithm for making something
blink.

2.

```
     ( START )
         ↓
  [ CLEAR SCREEN ] ←┐
         ↓          │
  [ FOR–NEXT        │
    TIME LOOP ]     │
         ↓          │
  [ PRINT           │
    "WOWSERS" ] ────┘
```
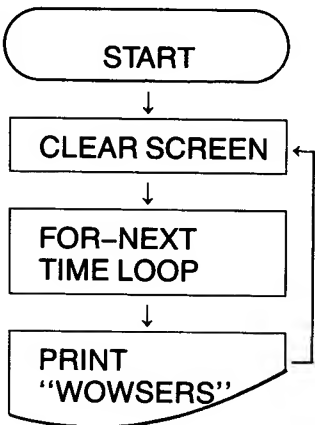
10 ? " [ESC] [SHIFT] [CLEAR <] "
20 FOR T = 1 TO 100:
   NEXT T
30 ? "WOWSERS"
40 GOTO 10

NOTE:
The time loops in each
program may vary.

Use your expertise and your imagination to write two of your own programs that make something blink. You can even make graphics or pictures blink! Don't be afraid to experiment.

Flow Chart   **Program**

1.

2.

Answers will vary.

COMPONENT 5 FUN PAGE

Answers

```
                                        7
                                        S
                                        A
          1                             V
          C                           6 5
        2 B U G                       S T E P
          O                             I
        4 N E X T                       M
          N                             E
          T
          E
          R

                    8 3
                    C L E A R                    11
                  9   O                          B
                  L   O                          L
                 10 I                            I
                  S T O P                        N
                  T                         12 B R E A K
```

133

Answers

Study each program and write what you think ATARI would print as the output. Run the programs to check your answers.

Program   **Output**

1. 5 DIM F$(10)   44
    10 LET F$ = "44"   44
    20 LET F = 44
    30 ? F$: ? F
    40 END

2. 5 DIM G$(10)   6+32=38
    10 LET G$ = "6+32="
    20 LET G = 6+32
    30 ? G$; G
    40 END

3. 5 DIM A$(10), S$(12)   ADDITION   SUBTRACTION
    10 LET A$ = "ADDITION"   8+8=16   8−8=0
    20 LET S$ = "SUBTRACTION"
    30 ?A$, S$
    40 LET A = 8+8
    50 LET S = 8−8
    60 ? "8+8="; A, "8−8="; S
    70 END

4. 5 DIM Q$(10), R$(10)
    10 LET Q$ = "HI HO"   HI HO   SILVER
    20 LET R$ = "SILVER!"   HI HO   SILVER
    30 ? Q$, R$ HI HO   SILVER
    40 GOTO 30   ⋮   ⋮

5. 5 DIM Z$(20)   YOU'RE OUTA SIGHT!
    10 LET Z$ = "YOU'RE OUTA SIGHT!"   YOU'RE OUTA SIGHT!
    20 FOR C = 1 TO 20   ⋮
    30 ? Z$ (20 times)
    40 NEXT C
    50 END

PROGRAMMER'S PASTIME #39

Answers

There is at least one mistake in each program (there may be more!). Find the mistake, circle the line number where you found it, then write the statement(s) the correct way in the space to the right.

Program

1. ⑤
   10 LET AZ$= "YES"
   20 LET BY$= "NO"
   30 ? AZ$, BY$
   40 END

2. 5 DIM T$(10), U$(10)
   10 LET T$= "THE TIME"
   20 LET U$= "IS NOW"
   ㉚ ? T, U
   40 END

3. 5 DIM J$(10), K$(10)
   ⑩ LET J$=UP, UP
   ⑳ LET K$=AND AWAY
   30 ? J$; ? K$
   40 END

4. 5 DIM P$(20), T$(20)
   ⑩ LET "PARTRIDGE IN" =P$
   ⑳ LET "A PEAR TREE" =T$
   30 ? P$, T$
   40 END

Correction

5 DIM AZ$(5),BY$(5)

30 ? T$, U$

10 LET J$= "UP, UP"
20 LET K$= "AND AWAY"

10 LET P$= "PARTRIDGE IN"
20 LET T$= "A PEAR TREE"

135

PROGRAMMER'S PASTIME #40

Answers

Study each program. Write what you think ATARI would print as the output. You may write what you would answer for each INPUT statement. Run the programs to check your work.

Program

```
1.  5 DIM A$(20)
   10 ? "HOW OLD ARE YOU";
   20 INPUT A
   30 ? "WHAT'S IT LIKE TO BEϼ";A;
   40 INPUT A$
   50 ? "I'M GLAD TO HEAR IT'Sϼ";
      A$
   60 END
```

Output

```
HOW OLD ARE YOU? 10
WHAT'S IT LIKE TO BE 10? OK
I'M GLAD TO HEAR IT'S OK
```

```
2.  5 DIM A$(20)
   10 ? "HOW OLD ARE YOU?"
   20 INPUT A
   30 ? "WHAT'S IT LIKE TO BEϼ";
      A; "?"
   40 INPUT A$
   50 ? "SO YOU AREϼ";
      A$; "ϼTODAY"
   60 END
```

```
HOW OLD ARE YOU?
? 9
WHAT'S IT LIKE TO BE 9?
? GREAT
SO YOU ARE GREAT TODAY
```

```
3. 10 ? "HOW MANY BROTHERS AND"
   20 ? "SISTERS DO YOU HAVE?"
   30 ? "TYPE NUMBER OF BROTHERS,"
   40 ? "A COMMA,"
   50 ? "AND NUMBER OF SISTERS"
   60 INPUT B, S
   70 LET T = B+S
   80 ? "SO YOU HAVEϼ"; T;
      "ϼSIBLINGS"
   90 END
```

```
HOW MANY BROTHERS AND
SISTERS DO YOU HAVE?
TYPE NUMBER OF BROTHERS,
A COMMA,
AND NUMBER OF SISTERS
? 4,5
SO YOU HAVE 9 SIBLINGS
```

136

Program	Output
4. 10 ?"CHOOSE TWO NUMBERS AND"	CHOOSE TWO NUMBERS AND I WILL ADD THEM FOR YOU
20 ?"I WILL ADD THEM FOR YOU"	TYPE 1ST NUMBER, COMMA,
30 ?"TYPE 1ST NUMBER, COMMA,"	THEN TYPE THE 2ND NUMBER
40 ?"THEN TYPE THE 2ND NUMBER"	?48,89
50 INPUT F, S	
60 ?	48+89=137
70 ?F;"+";S;"=";F+S	
80 ?	I'M A WHIZ!
90 ?"I'M A WHIZ!"	
100 END	
5. 10 ?"TYPE IN 2 NUMBERS"	TYPE IN 2 NUMBERS
20 ?"SEPARATED BY A COMMA"	SEPARATED BY A COMMA
30 INPUT O, T	?16,166
40 ?	
50 ?O;"+";T;"=";O+T	16+166=182
60 ?O;"−";T;"=";O−T	16−166=−150
70 ?O;"*";T;"=";O*T	16*166=2656
80 ?O;"/";T;"=";O/T	16/166=.0963855422
90 ?"SEE...I TOLD YOU"	SEE...I TOLD YOU
100 ?"I WAS A WHIZ!"	I WAS A WHIZ!
110 END	

PROGRAMMER'S PASTIME #41

Answers

Write a program for each flow chart. Run your programs on ATARI to check for bugs.

Flow Chart

1.

START
↓
DIMENSION VARIABLE
↓
PRINT "WHAT'S YOUR NAME"
↓
INPUT N$
↓
PRINT "HI∅"; N$
↓
STOP
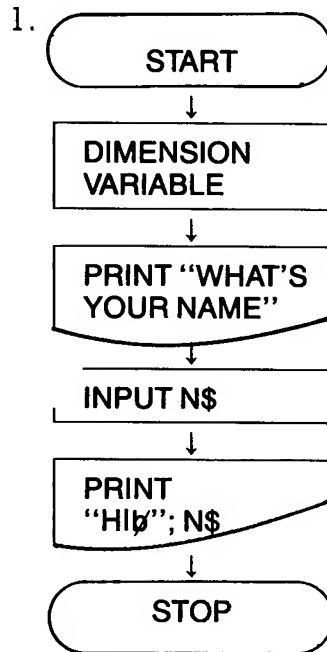
Program

```
10 DIM N$(20)
20 ? "WHAT'S YOUR
   NAME";
30 INPUT N$
40 ? "HI∅";N$
50 END
```

Flow Chart

2.

```
      ( START )
          ↓
   [ CLEAR SCREEN ]
          ↓
   PRINT "TYPE 3
   NUMBERS"
          ↓
   PRINT
   "SEPARATED
   BY COMMAS"
          ↓
   INPUT
   OE,TW,TH
          ↓
   PRINT
   "OE;"+";TW;
   "+";TH;"=";
   OE+TW+TH
          ↓
   [ SKIP A LINE ]
          ↓
   PRINT
   "THANK YOU!"
          ↓
      ( STOP )
```
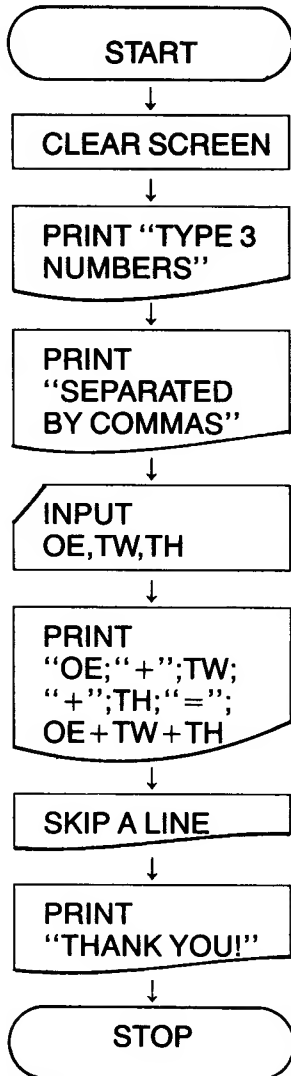
Program

10 ?`` [ESC] [SHIFT] [CLEAR <] ''
20 ?``TYPE 3
   NUMBERS''
30 ?``SEPARATED BY
   COMMAS''
40 INPUT OE, TW, TH
50 ?OE;``+'';TW;``+'';
   TH;``='';OE+TW+TH
60 ?
70 ?``THANK YOU!''
80 END

139

Flow Chart

3.

```
       START
         ↓
   DIMENSION
   VARIABLE
         ↓
   PRINT "WHAT
   IS YOUR
   ADDRESS"
         ↓
   INPUT A$
         ↓
   SKIP A LINE
         ↓
   PRINT
   "YOU LIVE
   ATØ";A$
         ↓
        STOP
```
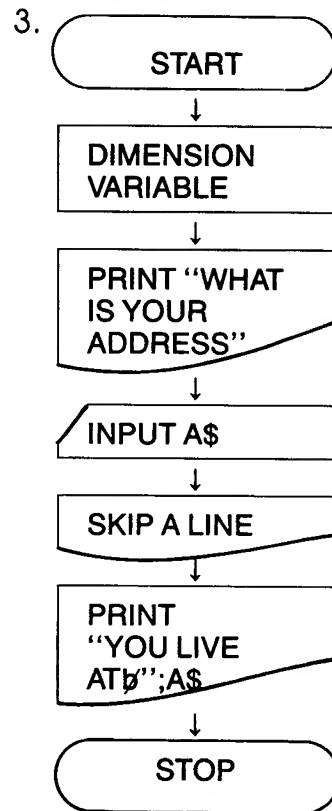
Program

```
10 DIM A$(20)
20 ? "WHAT IS YOUR
   ADDRESSØ";
30 INPUT A$
40 ?
50 ? "YOU LIVE ATØ";
   A$
60 END
```

Write 3 programs using the INPUT statement. Write the flow chart for the algorithm first, then write the program. Debug your programs by running them on ATARI.

Flow Chart   **Program**

Programs will vary

PROGRAMMER'S PASTIME #42

Answers

Write each equation as an IF–THEN statement.

Question	IF–THEN statement
Example:	
Is A equal to C?	IF A=C THEN _____
1. Is L$ equal to "MAYBE"?	IF L$= "MAYBE" THEN
2. Is F1 not equal to FZ?	IF F1 < > FZ THEN
3. Is GH greater than HI?	IF GH > HI THEN
4. Is S$ less than or equal to F$?	IF S$< = F$ THEN
5. Is X times B less than P times Q?	IF X*B<P*Q THEN
6. Is T divided by W greater than or equal to W times B?	IF T/W > =W*B THEN
7. Is P$ greater than M$?	IF P$>M$ THEN
8. Is the square root of Y equal to D?	IF SQR(Y)=D THEN
9. Is G$ not equal to "NO"?	IF G$ < > "NO" THEN
10. Is 10 divided by 5 less than 14 divided by 2?	IF 10/5 < 14/2 THEN
11. Is Y$ equal to the square root of 64?	IF Y$=SQR(64) THEN
12. Is A plus B greater than D$?	IF A+B> D$ THEN

Answers

For each question write the Complement
IF–THEN statement.

Question **Example**	**Complement** **IF–THEN statement**
Is P$ equal to "YES"?	IF P$ < > "YES" THEN _____
1. Is QR greater than 2?	IF QR< 2 THEN
2. Is Z$ not equal to "END"?	IF Z$ = "END" THEN
3. Is F less than or equal to P?	IF F> = P THEN
4. Is G$ equal to "JEEPERS"?	IF G$< > "JEEPERS" THEN
5. Is S1 greater than or equal to S2?	IF S1 < =S2 THEN
6. Is DD less than 444?	IF DD> 444 THEN
7. Is X greater than Y?	IF X< Y THEN
8. Is A$ greater than or equal to 79?	IF A$< =79 THEN
9. Is P$ not equal to "YES"?	IF P$= "YES" THEN
10. Is VP less than or equal to JK?	IF VP> =JK THEN

Answers

The location of the IF–THEN statement in a program is very important. If it is put in the wrong place, the program won't work properly. The IF–THEN statement must come **after** the LET or INPUT statements that introduce the variables in the IF–THEN statement. For example:

Program	Output
10 IF P < Q THEN 50	ATARI does not print
20 LET P=5	anything because the
30 LET Q=7	IF–THEN statement is
40 GOTO 60	before the LET state-
50 ? "P IS SMALLER"	ments that introduce
60 END	the variables.

In the following programs, the IF–THEN statement is in the wrong place. Rewrite the programs so they are correct.

Incorrect program

1. 10 IF Z=2 THEN 60
   20 LET A=6
   30 LET B=8
   40 LET Z=2
   50 GOTO 70
   60 ? "Z=2"
   70 END

Corrected program

10 LET A=6
20 LET B=8
30 LET Z=2
40 IF Z=2 THEN 60
50 GOTO 70
60 ? "Z=2"
70 END

Incorrect program	Corrected program
2. 5 DIM E$(10),D$(10)	5 DIM E$(10),D$(10)
10 ?"WHAT COLOR ARE YOUR EYES?"	10 ?"WHAT COLOR ARE YOUR EYES?"
20 IF E$="BLUE" THEN 100	20 INPUT E$
30 INPUT E$	30 IF E$="BLUE" THEN 100
40 ?"ARE THEY DIFFERENT COLORS?"	40 ?"ARE THEY DIFFERENT COLORS?"
50 IF D$="YES" THEN 120	50 INPUT D$
60 INPUT D$	60 IF D$="YES" THEN 120
70 ?"THEY ARE 1 COLOR"	70 ?"THEY ARE 1 COLOR"
80 ?"THEY ARE NOT BLUE"	80 ?"THEY ARE NOT BLUE"
90 GOTO 130	90 GOTO 130
100 ?"WHAT NICE BLUE EYES!"	100 ?"WHAT NICE BLUE EYES!"
110 GOTO 130	110 GOTO 130
120 ?"WHAT COLORFUL EYES!"	120 ?"WHAT COLORFUL EYES!"
130 END	130 END
3. 5 DIM F$(10)	5 DIM F$(10)
10 IF F$="NO" THEN 60	10 ?"ARE COMPUTERS FUN?"
20 ?"ARE COMPUTERS FUN?"	20 INPUT F$
30 INPUT F$	30 IF F$="NO" THEN 60
40 ?"YOU'RE RIGHT!"	40 ?"YOU'RE RIGHT!"
50 GOTO 70	50 GOTO 70
60 ?"YOU'RE NO FUN!"	60 ?"YOU'RE NO FUN!"
70 END	70 END

PROGRAMMER'S PASTIME #45

Answers

Study each flow chart, then write a program.
Debug your programs by running them on ATARI.

Flow Chart

1.

```
      ( START )
          ↓
   ┌──────────────┐
   │ DIMENSION    │
   │ VARIABLE     │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ PRINT        │
   │ "WHEN IS     │
   │ YOUR         │
   │ BIRTHDAY?"   │
   └──────────────┘
          ↓
   ┌──────────────┐
  /  INPUT B$     │
   └──────────────┘
          ↓
       ╱DOES╲
      ╱ B$ =  ╲────YES
      ╲"DECEMBER"╱
       ╲  ?  ╱
          ↓ NO
   ┌──────────────┐
   │ PRINT "YOUR  │
   │ BIRTHDAY IS  │
   │ NOT IN       │
   │ DECEMBER"    │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ GOTO STOP    │
   └──────────────┘
          ↓
   ┌──────────────┐
   │ PRINT "YOUR  │
   │ BIRTHDAY IS IN│
   │ THE LAST     │
   │ MONTH OF THE │
   │ YEAR"        │
   └──────────────┘
          ↓
      ( STOP )
```
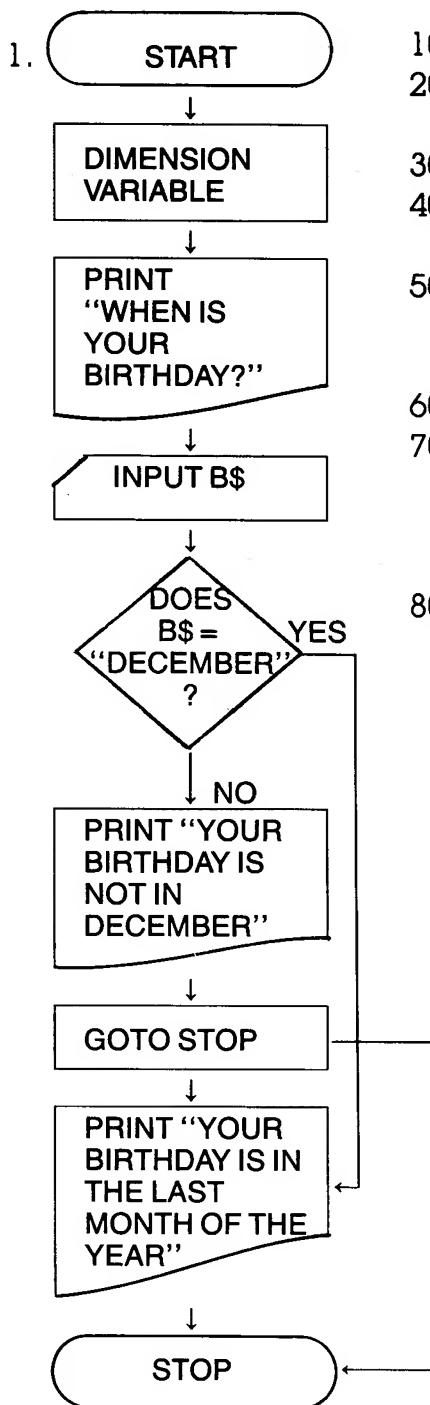
Program

```
10 DIM B$(20)
20 ? "WHEN IS YOUR
   BIRTHDAY";
30 INPUT B$
40 IF B$ = "DECEMBER"
   THEN 70
50 ? "YOUR BIRTHDAY
   IS NOT IN
   DECEMBER"
60 GOTO 80
70 ? "YOUR BIRTHDAY
   IS IN THE LAST
   MONTH OF THE
   YEAR"
80 END
```

146

Flow Chart

2.

```
     ( START )
         ↓
   ┌──────────────┐
   │  DIMENSION   │
   │  VARIABLES   │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ PRINT        │
   │ "HELLO"      │
   │ "WHAT IS     │
   │ YOUR NAME?"  │
   └──────────────┘
         ↓
   ┌──────────────┐
   │  INPUT N$    │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ SKIP         │
   │ A LINE       │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ PRINT        │
   │ "HOW ARE YOU │
   │ TODAY?"      │
   └──────────────┘
         ↓
   ┌──────────────┐
   │  INPUT H$    │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ SKIP         │
   │ A LINE       │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ PRINT "SINCE │
   │ YOU ARE";H$; │
   │ "TODAY";N$   │
   └──────────────┘
         ↓
   ┌──────────────┐
   │ PRINT "WE    │
   │ MUST GET     │
   │ TO WORK"     │
   └──────────────┘
         ↓
     ( STOP )
```
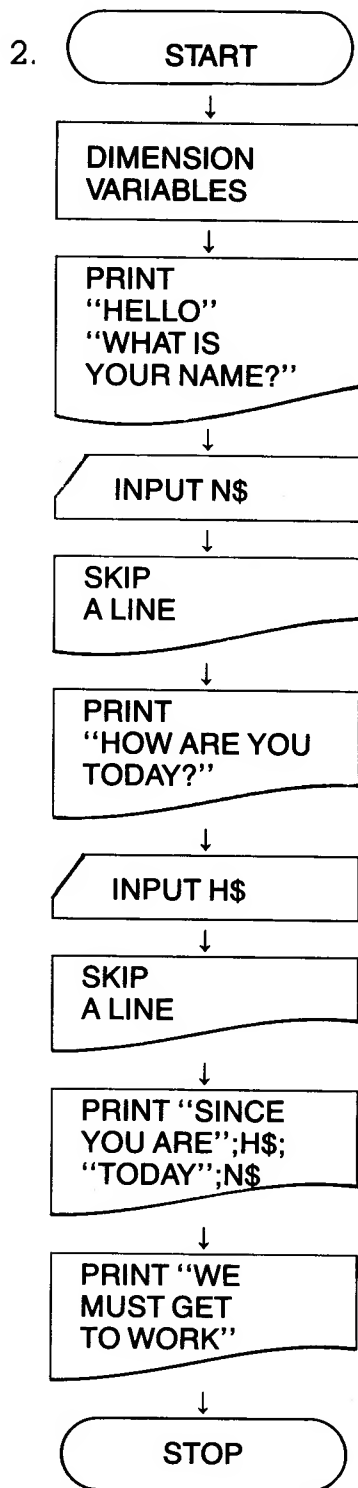
Program

```
10  DIM N$(35), H$(20)
20  ? "HELLO."
30  "WHAT'S YOUR
    NAME";
40  INPUT N$
50  ?
60  ? "HOW ARE YOU
    TODAY";
70  INPUT H$
80  ?
90  ? "SINCE YOU
    AREb";H$;
    "bTODAYb";N$
100 ? "WE MUST GET
    TO WORK"
110 END
```

Flow Chart

3.

START

↓

DIMENSION
VARIABLES

↓

PRINT "HELLO
WHAT'S YOUR
NAME?"

↓

INPUT N$

↓

SKIP
A LINE

↓

PRINT "HOW
ARE YOU
TODAY? FINE
OR ROTTEN"

↓

INPUT H$

↓

DOES
H$ =
"ROTTEN"
? → NO

↓ YES

PRINT "I'M
SORRY";N$;
"THAT YOU
ARE" H$

↓

GOTO STOP

↓

PRINT
"GREAT";
N$;"LET'S
GET DOWN
TO WORK"

↓

STOP
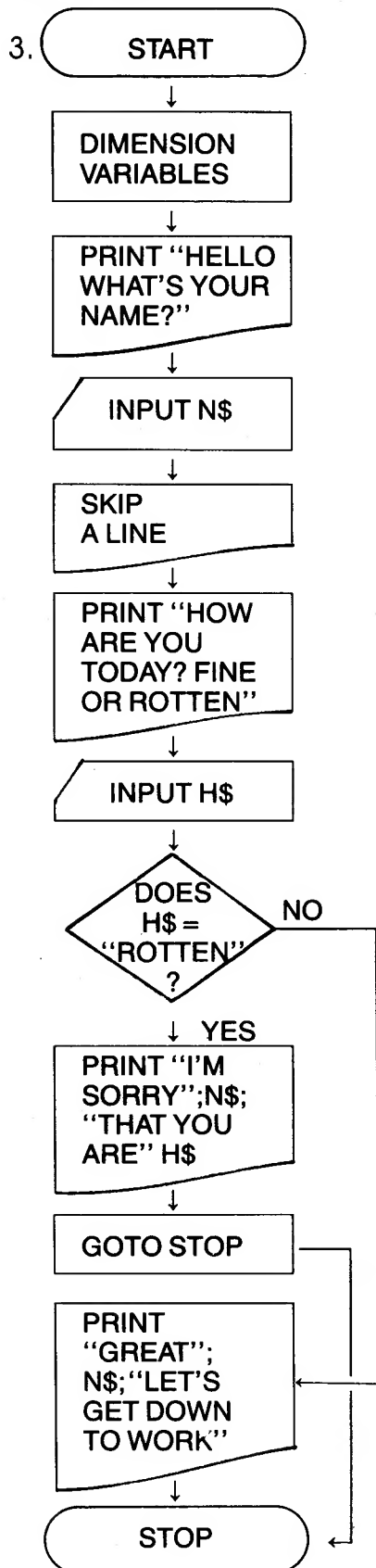
Program

```
10  DIM N$(35),H$(10)
20  ? "HELLO"
30  ? "WHAT'S YOUR
    NAME?";
40  INPUT N$
50  ?
60  ? "HOW ARE YOU
    TODAY?"
70  ? "FINE OR
    ROTTEN?";
80  INPUT H$
90  IF H$ < >
    "ROTTEN" THEN
    120
100 ? "I'M SORRYØ";
    N$;"ØTHAT YOU
    AREØ";H$
110 GOTO 140
120 ? "GREATØ";N$;
130 ? "ØLET'S GET
    DOWN TO
    WORK!"
140 END
```

Clue: You will need to use the complement of the question for your IF–THEN statement.

Flow Chart

4.

```
        ( START )
            ↓
   ┌─────────────────┐
   │ DIMENSION       │
   │ VARIABLES       │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ PRINT           │
   │ "WHAT IS        │
   │ YOUR NAME"      │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ INPUT N$        │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ SKIP            │
   │ A LINE          │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ PRINT           │
   │ "HI";N$;        │
   │ "DO YOU         │
   │ LIKE SCHOOL?"   │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ INPUT S$        │
   └─────────────────┘
            ↓
   ┌─────────────────┐
   │ SKIP            │
   │ A LINE          │
   └─────────────────┘
            ↓
         ◇ DOES
         S$ = "YES"  ── NO ──┐
           ?                  │
            │ YES             │
            ↓                 ↓
   ┌─────────────────┐  ┌─────────────────┐
   │ PRINT "GOOD     │  │ PRINT "AW       │
   │ FOR YOU!        │  │ C'MON—BE A      │
   │ SEE YA";N$      │  │ SPORTb";N$      │
   └─────────────────┘  └─────────────────┘
            ↓                 ↓
   ┌─────────────────┐     ( STOP )
   │ GOTO STOP       │  →
   └─────────────────┘
```
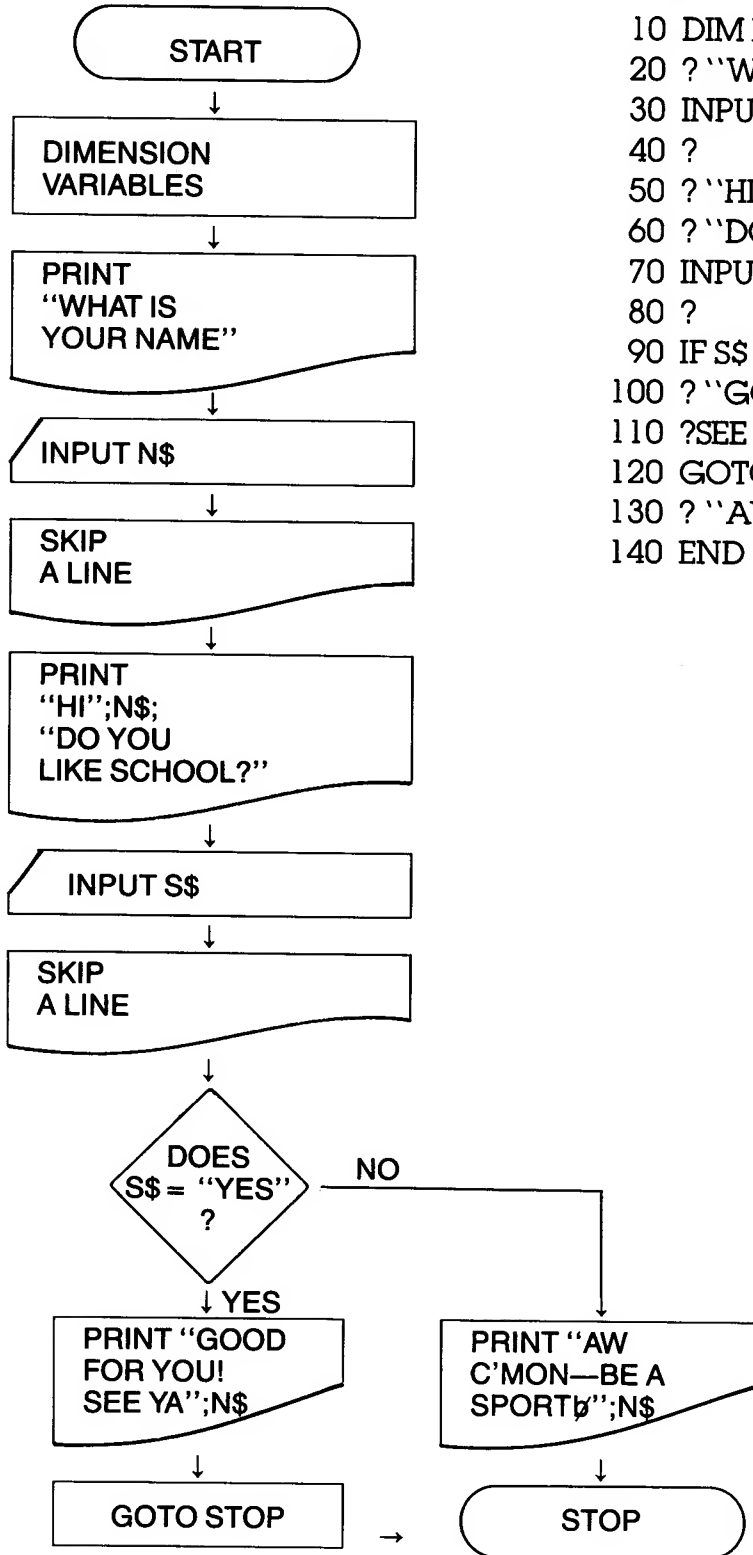
Program

```
 10 DIM N$(35),S$(35)
 20 ?"WHAT IS YOUR NAME";
 30 INPUT N$
 40 ?
 50 ?"HIb";N$
 60 ?"DO YOU LIKE SCHOOL";
 70 INPUT S$
 80 ?
 90 IF S$< >"YES" THEN 130
100 ?"GOOD FOR YOU!'
110 ?SEE YAb";N$
120 GOTO 140
130 ?"AW C'MON—BE A SPORTb";N$
140 END
```

Clue: You will need to use the complement of the question for your IF–THEN statement.

PROGRAMMER'S PASTIME #46

Answers

For each description, write an algorithm in flow chart form and write a program for the flow chart. Debug each program by running it on ATARI.

Description	Flow Chart	Program

Description
1. Alphabetize "HIP" and "HIPPO"

Flow Chart

```
        START
          ↓
    DIMENSION
    VARIABLES
          ↓
   LET A$ = "HIP"
   LET B$ = "HIPPO"
          ↓
         IS
       A$ < B$        YES
         ?
          ↓ NO
      PRINT
      B$ : A$
          ↓
    PRINT A$ : B$
          ↓
        STOP
```
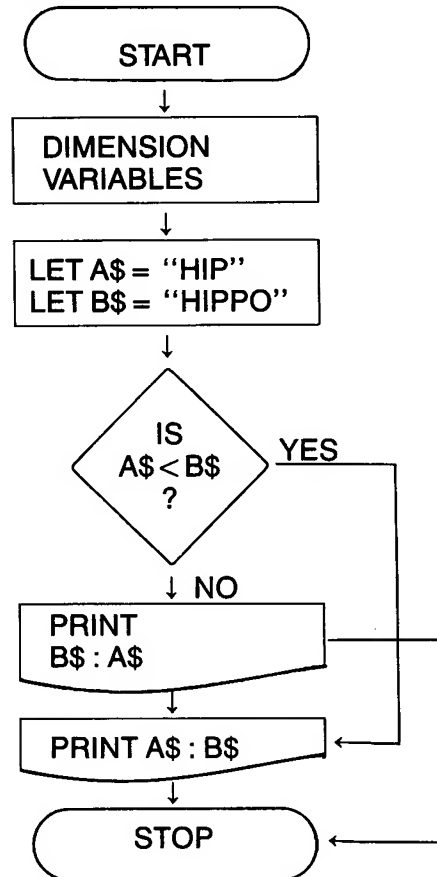
Program
```
 5 DIM A$(10),B$(10)
10 LET A$= "HIP"
20 LET B$= "HIPPO"
30 IF A$<B$ THEN 60
40 ?B$ :?A$
50 GOTO 70
60 ?A$ :?B$
70 END
```

2. Alphabetize "GUSTO" and "GROOVY"
   Flow chart is basically the same as #1

   LET A$= "GUSTO"
   LET B$= "GROOVY"

```
 5 DIM A$(10),B$(10)
10 LET A$= "GUSTO"
20 LET B$= "GROOVY"
30 IF A$<B$ THEN 60
40 ?B$ :?A$
50 GOTO 70
60 ?A$ :?B$
70 END
```

150

DESCRIPTION	FLOW CHART	PROGRAM

3. Alphabetize
   "AARDVARK" and
   "ZEBRA"
   Flow chart is basically
   the same as #1
   LET A$ = "AARDVARK"
   LET B$ = "ZEBRA"

```
 5 DIM A$(10),B$(10)
10 LET A$=
   "AARDVARK" : LET
   B$= "ZEBRA"
20 IF A$<B$ THEN 50
30 ? B$ : ? A$
40 GOTO 60
50 ? A$ : ? B$
60 END
```

PROGRAMMER'S PASTIME #47

Answers

Study each flow chart and then write a program. Use REM statements where appropriate to show good programming style. Debug your programs by running them on ATARI.

Flow Chart

1.
START
↓
DIMENSION VARIABLE
↓
LET Q$ = "WHOOPEE"
↓
FOR C = 1 TO 10
↓
PRINT C
↓
PRINT Q$
↓
NEXT C
↓
PRINT "DONE"
↓
STOP
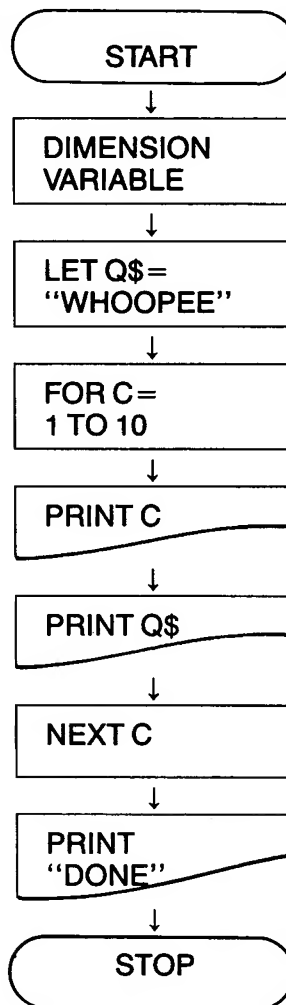
Program

```
10 DIM Q$(10)
20 REM PRINTING Q$
   10 TIMES
30 LET Q$=
   "WHOOPEE"
40 FOR C=1 TO 10
50 ? C
60 ? Q$
70 NEXT C
80 ? "DONE"
90 END
```

Flow Chart

2.

```
        ┌──────────────┐
        │    START     │
        └──────────────┘
               ↓
        ┌──────────────┐
        │ CLEAR SCREEN │
        └──────────────┘
               ↓
        ┌──────────────┐
        │  DIMENSION   │
        │  VARIABLE    │
        └──────────────┘
               ↓
        ┌──────────────┐
        │ PRINT "WHAT  │
        │ IS YOUR      │
        │ FAVORITE     │
        │ COLOR?"      │
        └──────────────┘
               ↓
        ┌──────────────┐
        │  INPUT C$    │
        └──────────────┘
               ↓
            ╱ DOES ╲
           ╱  C$ =  ╲  YES
           ╲ "RED"  ╱────────┐
            ╲  ?  ╱          │
               ↓ NO          │
        ┌──────────────┐     │
        │ PRINT "YOU   │     │
        │ DON'T LIKE   │     │
        │ RED?         │     │
        │ RED IS MY    │     │
        │ FAVORITE     │     │
        │ COLOR"       │     │
        └──────────────┘     │
               ↓             │
        ┌──────────────┐     │
    ┌───│  GOTO STOP   │     │
    │   └──────────────┘     │
    │          ↓             │
    │   ┌──────────────┐     │
    │   │  PRINT       │←────┘
    │   │  "I LOVE     │
    │   │  RED TOO!"   │
    │   └──────────────┘
    │          ↓
    │   ┌──────────────┐
    └──→│    STOP      │
        └──────────────┘
```
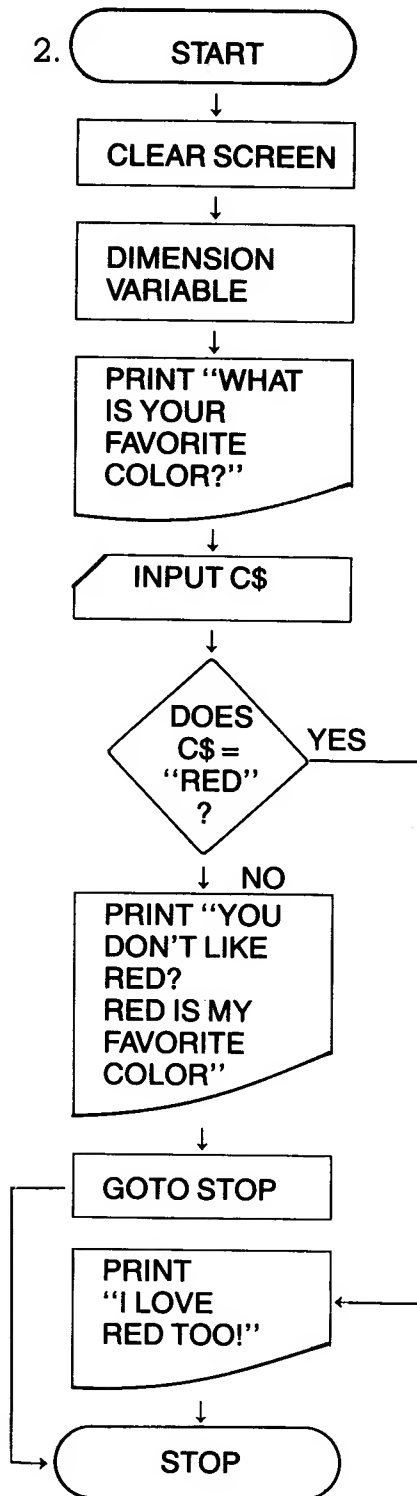
Program

```
 10  REM SETTING UP
     SCREEN
 20  ? " [ESC] [SHIFT] [CLEAR<] "
 30  DIM C$(15)
 40  REM FAVORITE
     COLOR
 50  ? "WHAT'S YOUR
     FAVORITE
     COLOR";
 60  INPUT C$
 70  IF C$ = "RED"
     THEN 110
 80  ? "YOU DON'T LIKE
     RED?"
 90  ? "RED IS MY
     FAVORITE
     COLOR"
100  GOTO 120
110  ? "I LOVE RED
     TOO!"
120  END
```

153

Flow chart

3.

```
        START
          ↓
    CLEAR SCREEN
          ↓
       PRINT
     "60+40=";
          ↓
      INPUT S
          ↓
       DOES        YES
      S=100  ─────────→
        ?
          ↓ NO
    PRINT "NO.
    TRY AGAIN"
          ↓
       GOTO
          ↓
       PRINT
    "GOOD JOB!"  ←────
          ↓
       PRINT
     "78-14=";  ←──────────
          ↓
      INPUT D
          ↓
       DOES        NO        PRINT
      D=64   ─────────→      "NO
        ?                  TRY AGAIN"
          ↓ YES                ↓
       PRINT                  GOTO  ──────┘
      "SUPER"
          ↓
       STOP
```
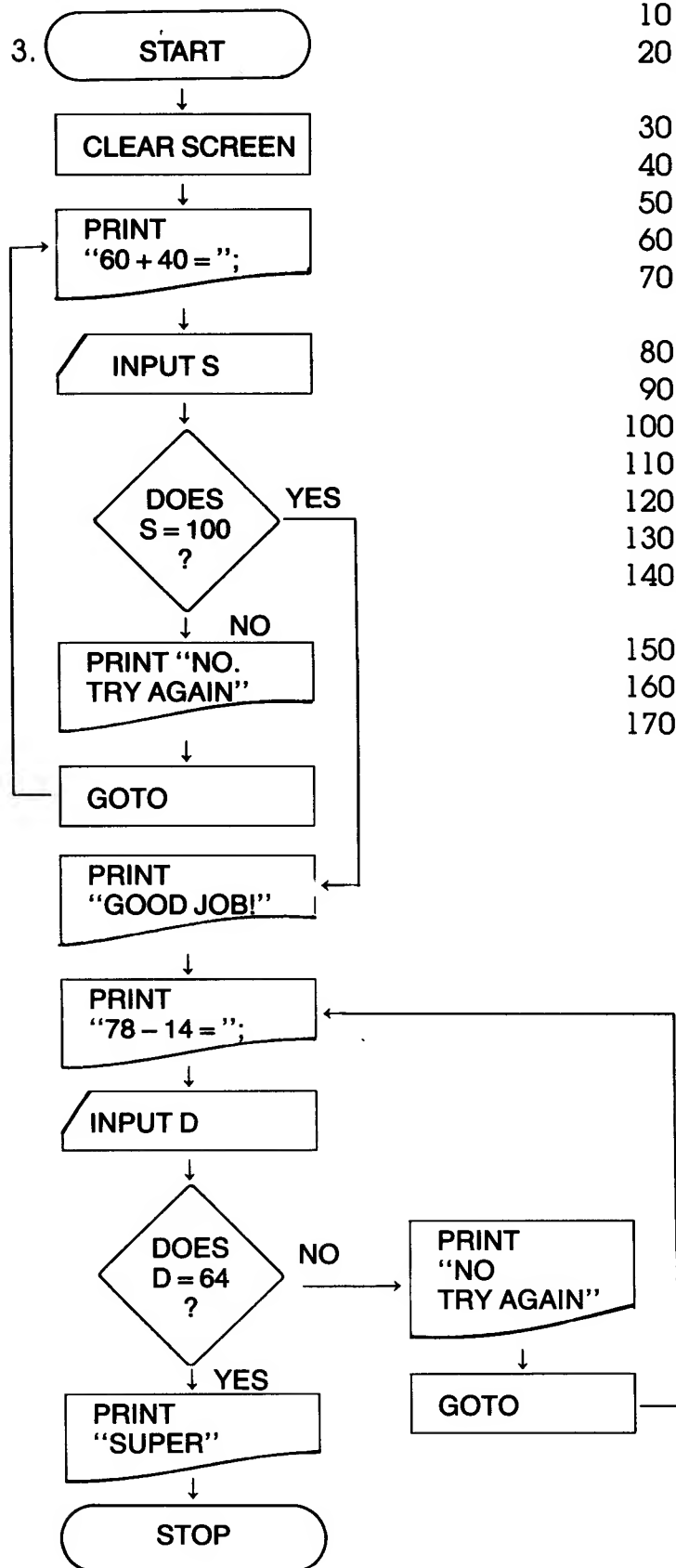
Program

10 ? `` [ESC] [SHIFT] [CLEAR <] ``

20 REM MATH
    PRACTICE

30   REM ADDITION

40 ? "60+40=";

50 INPUT S

60 IF S= 100 THEN 90

70 ? "NO. TRY
    AGAIN."

80 GOTO 30

90 ? "GOOD JOB!"

100   REM SUBTRACTION

110 ? "78−14=";

120 INPUT D

130 IF D=64 THEN 160

140 ? "NO. TRY
    AGAIN"

150 GOTO 110

160 ? "SUPER"

170 END

PROGRAMMER'S PASTIME #48

Study each program and write what you think
ATARI would print as the OUTPUT—including error
messages. Check your answers by running the
programs on ATARI.

Program

Output

```
1.  5 DIM Z$(10),L$(10)    YOU         ARE
   10 READ Z$, L$          A           HOT-SHOT
   20 ? Z$, L$             ERROR—6 AT LINE 10
   30 GOTO 10
   40 DATA "YOU",
      "ARE", "A",
      "HOT-SHOT"
   50 END
```

```
2. 10 DATA 4,14,41,6,       59
      16,61,3,13,31          83
   20 READ Q, R, S           47
   30 ? Q+R+S                ERROR—6 AT LINE 20
   40 GOTO 20
   50 END
```

```
3. 10 READ G, H             44          66
   20 DATA 44,66,88,        88          22
      22,110                ERROR—6 AT LINE 10
   30 ? G, H
   40 GOTO 10
   50 END
```

```
4. 10 DATA 14,7,2,16,8,     5
      2,-99,-99,-99          6
   20 READ A, L, B
   30 IF A=-99 THEN
      60
   40 ? A, L, B
   50 GOTO 20
   60 END
```

Program	Output	
5. 10 READ R1, R2	4	
20 IF R1 = -1 THEN	9	
60	16	
30 ? R1*R2	25	
40 GOTO 10		
50 DATA 2,2,3,3,4,		
4,5,5,-1,-1		
60 END		

Program	Output	
6. 5 DIM D$(10),E$(10)	A	E
10 FOR L=1 TO 4	I	O
20 READ D$, E$	U	Y
30 ? D$, E$	ARE	VOWELS
40 NEXT L		
50 DATA "A", "E",		
"I", "O"		
60 DATA "U", "Y",		
"ARE",		
"VOWELS"		
70 END		

Program	Output	
7. 10 FOR L=1 TO 2	64	
20 READ S1, S2, S3	36	
30 DATA 8,2,4,6,2,3		
40 ? S1*S2*S3		
50 NEXT L		
60 END		

Answers

In each of the following programs there are mistakes. Circle the line number(s) with the mistake and make your correction in the space to the right. If something has been left out, add it to the program.

Program   **Correction**

1. 10 READ P,A,N
   20 ?P,A,N
   ⓃⒶ DATA 400, 8%, 6   30 DATA 400, 8, 6
   40 END

2. 5 DIM N$(10)
   10 READ N$, A
   20 ?N$, A
   ㉚ DATA KIM IS,   30 DATA KIM IS , 12
      3*4
   40 END

3. 5 DIM N$(10)
   10 ? "NAME",
      "AGE"
   ⑳ READ A, N$ 20 READ N$, A
   ㉚ ? A, N$ 30 ?N$, A
   40 DATA HARVEY,
      14
   50 END

4. 5 DIM N$(10)
   10 ? "NAME", "AGE"
   20 READ N$, A
   30 ?N$, A
   ㊵ DATA HARVEY,   40 DATA HARVEY , 14
      14 YEARS OLD
   50 END

157

Program	Correction
5. 5 DIM F$(20)	(No DATA statement)
10 READ F$	15 DATA FRIED FISH
20 ? ''DAILY MENU''	
30 ? F$	
40 END	
6. 10 READ X,Y,Z	(Not enough data)
20 ? ''THE PRODUCT OF 3 NUMBERS''	
30 ? X*Y*Z	
40 DATA 4,5	40 DATA 4,5,6
50 END	
7. 10 ? ''COUNTING''	
20 READ DATA	20 READ A, B, C, D
30 DATA 1,2,3,4	25 ? A, B, C, D
40 END	
8. 5 DIM N$(10)	
10 ? ''NAME'', ''AGE''	
20 READ N$, A	
30 ? N$, A	
40 GOTO 20	
50 DATA BOB, BILL,10,11	50 DATA BOB, 10, BILL, 11

PROGRAMMER'S PASTIME #50

READ–DATA statements can help you write shorter programs. Rewrite each program using READ–DATA statements to shorten them. Try to write each program so you don't get an error message.

Long program

```
1. 10 ? "MULTIPLYING
       2 NUMBERS"
   20 LET P=60
   30 LET Q=129
   40 LET R=410
   50 LET S=.6
   60 ?P,Q, P*Q
   70 ?R,S, R*S
   80 END
```

Short program

```
10 ? "MULTIPLYING 2
      NUMBERS"
20 READ P,Q,R,S
30 DATA 60, 129,
      410, .6
40 ?P, Q, P*Q:
      ?R, S, R*S
50 END
```

```
2.   5 DIM A$(10),
       B$(10), C$(10),
       D$(10), E$(10)
    10 ? "TEST SCORES"
    20 ? "NAME",
       "SCORE"
    30 LET A$= "JOE"
    40 LET A=98
    50 LET B$= "TOM"
    60 LET B=52
    70 LET C$= "KRIS"
    80 LET C=95
    90 LET D$= "GAIL"
   100 LET D=75
   110 LET E$= "BOB"
   120 LET E=72
   130 ?A$, A
   140 ?B$, B
   150 ?C$, C
   160 ?D$, D
   170 ?E$, E
   180 END
```

```
 5 DIM N$(10)
10 ? "TEST SCORES"
20 ? "NAME",
      "SCORE"
30 FOR C=1 TO 5
40 READ N$,S
50 ?N$,S
60 NEXT C
70 DATA JOE, 98,
      TOM, 52, KRIS, 95,
      GAIL, 75, BOB, 72
80 END
```

PROGRAMMER'S PASTIME #51

Answers

Study each flow chart. Then write a program using a READ–DATA statement. Use any DATA that you feel will work in the DATA statement. Debug your programs by running them on ATARI.

Flow chart

Program

1.
START

↓

PRINT
"FINDING AREA"

↓

PRINT
"LENGTH",
"WIDTH",
"AREA"

↓
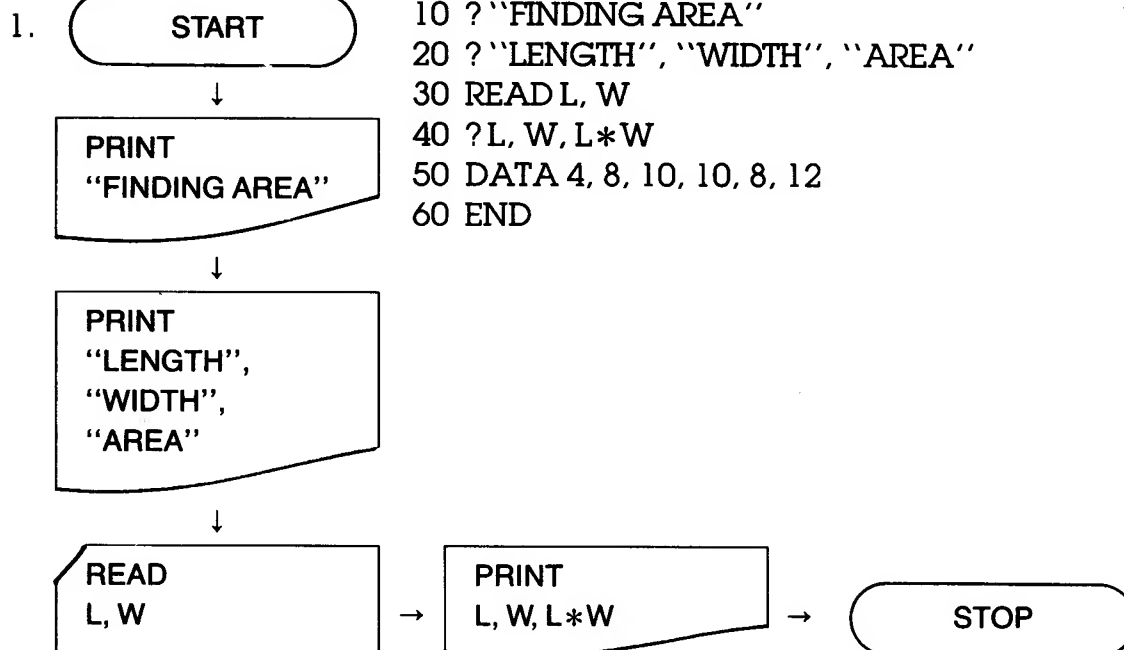
READ
L, W   →   PRINT
L, W, L*W   →   STOP

```
10 ? "FINDING AREA"
20 ? "LENGTH", "WIDTH", "AREA"
30 READ L, W
40 ? L, W, L*W
50 DATA 4, 8, 10, 10, 8, 12
60 END
```

Flow chart	Program

2.

```
START
  ↓
PRINT
"FINDING
PERIMETER"
  ↓
FOR X = 1 TO 3  ←──────────┐
  ↓                        │
READ          →  PRINT        →  NEXT X
S1, S2, S3, S4   S1, S2, S3, S4,        │
                 S1 + S2 + S3 + S4      ↓
                                     STOP
```
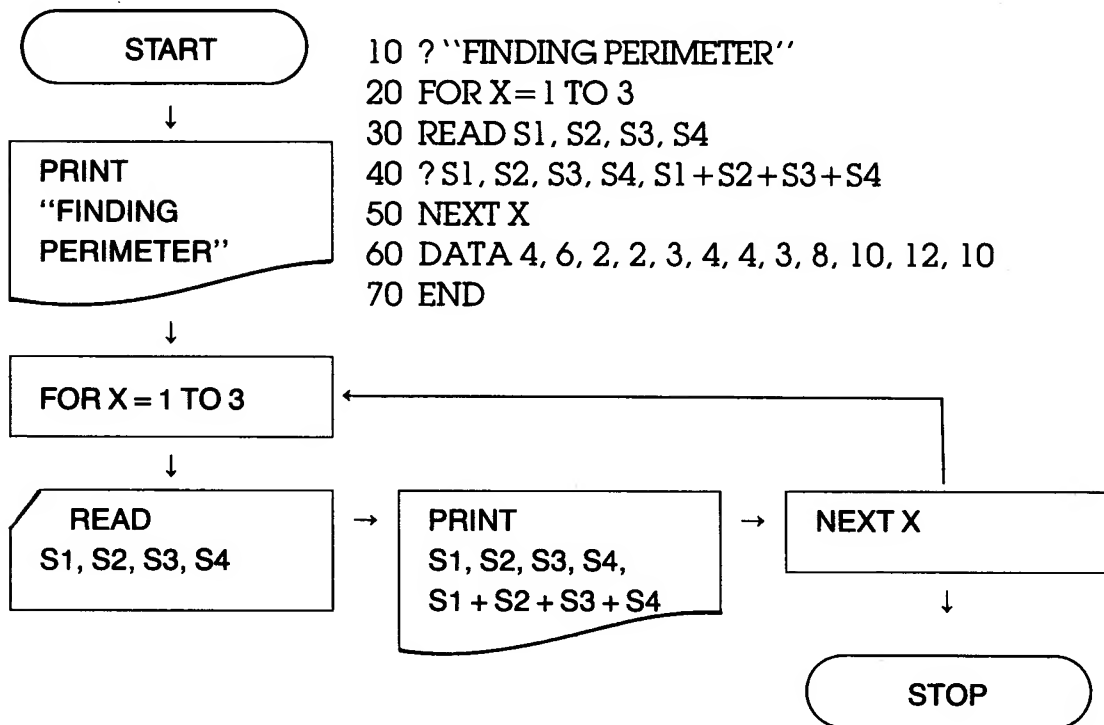
```
10 ? "FINDING PERIMETER"
20 FOR X = 1 TO 3
30 READ S1, S2, S3, S4
40 ? S1, S2, S3, S4, S1+S2+S3+S4
50 NEXT X
60 DATA 4, 6, 2, 2, 3, 4, 4, 3, 8, 10, 12, 10
70 END
```

3.

```
START
  ↓
PRINT
"DIVIDING
NUMBERS"
  ↓
READ          ←──────────┐
D1, D2                   │
  ↓                      │
  ╱IS  ╲   YES           │
 ◄ D1 = -9 ►──────────┐  │
  ╲  ?  ╱             │  │
  ↓ (no)             │  │
PRINT      →  GOTO    →  STOP
D1, D2, D1/D2
```
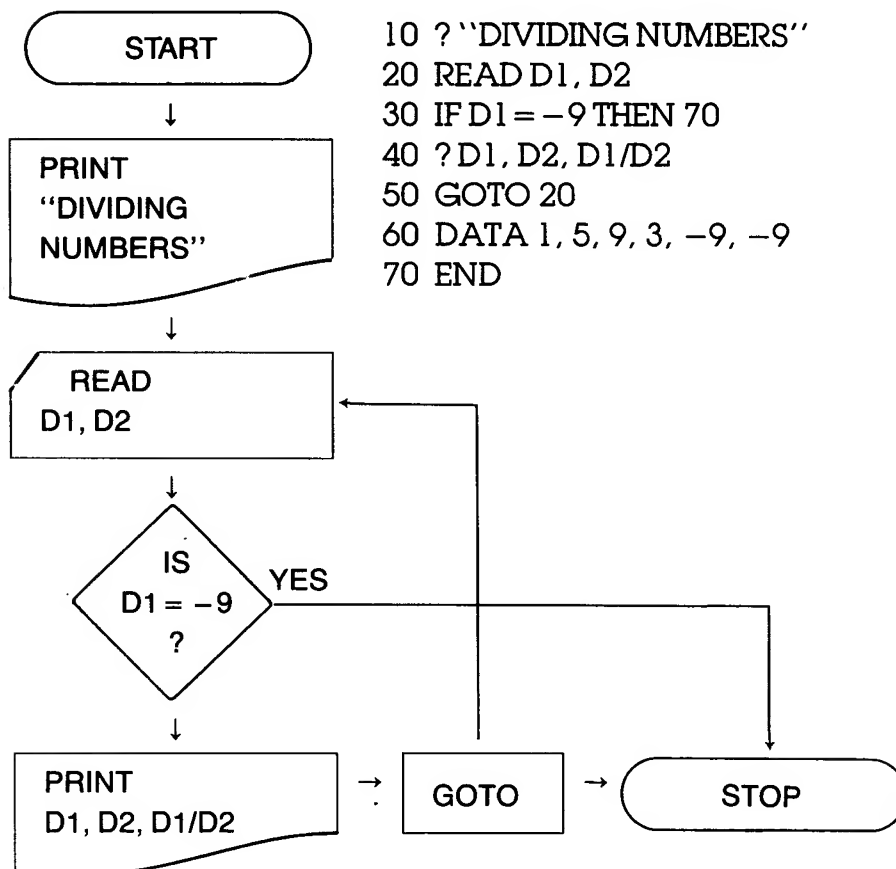
```
10 ? "DIVIDING NUMBERS"
20 READ D1, D2
30 IF D1 = -9 THEN 70
40 ? D1, D2, D1/D2
50 GOTO 20
60 DATA 1, 5, 9, 3, -9, -9
70 END
```

4.

```
START
  ↓
PRINT "FINDING
THE AREA OF
A TRIANGLE"
  ↓
FOR L = 1 TO 3 ←
  ↓
READ
B,H
  ↓
PRINT B, H,     →   NEXT L   →   STOP
½*B*H
```
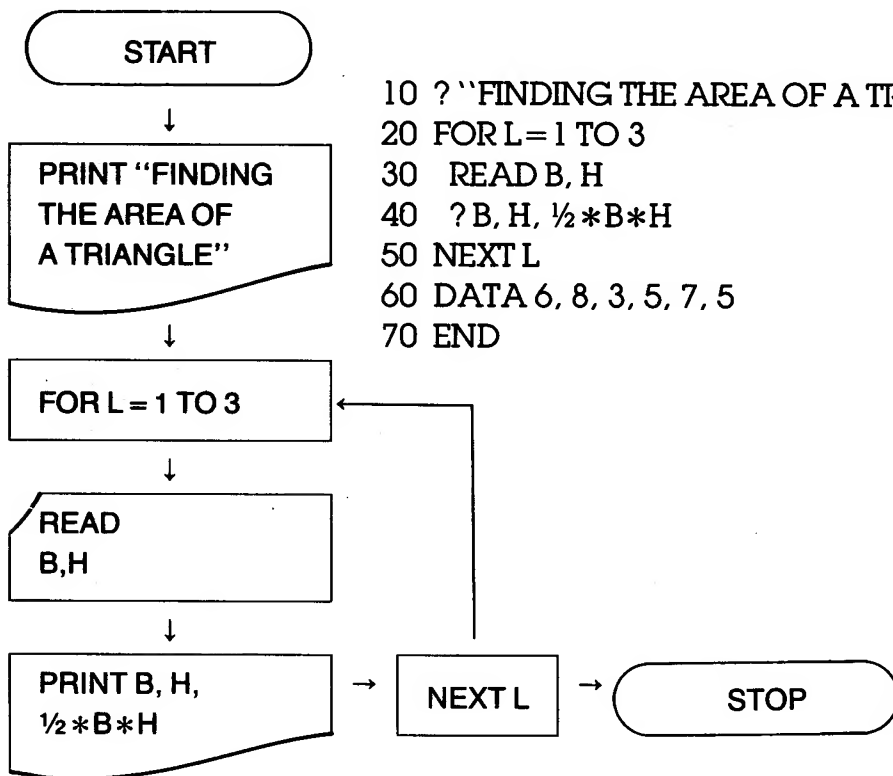
```
10  ?"FINDING THE AREA OF A TRIANGLE"
20  FOR L=1 TO 3
30    READ B, H
40    ? B, H, ½*B*H
50  NEXT L
60  DATA 6, 8, 3, 5, 7, 5
70  END
```

Answers

Using what you know about READ–DATA statements:

1. Write a program that multiplies three numbers.

(samples)

Flow chart
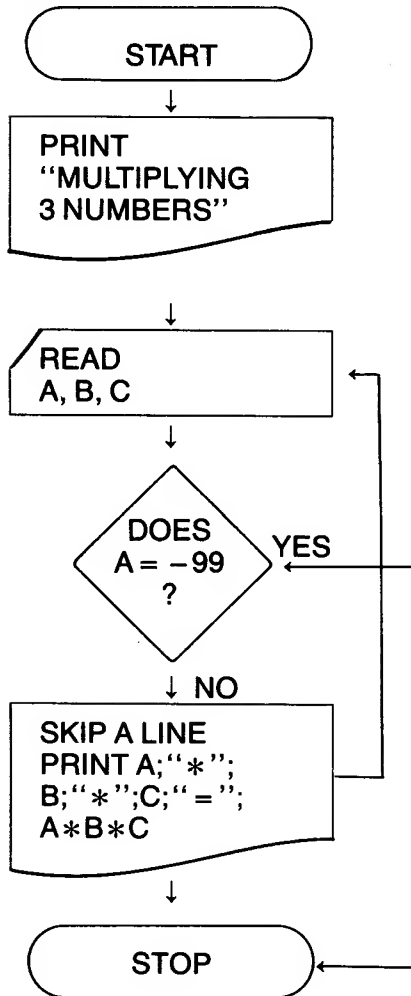
Program

```
10   ? "MULTIPLYING 3 NUMBERS"
20   READ A, B, C
30   IF A = -99 THEN 70
40   ? : ? A; " * "; B; " * "; C; " = "; A*B*C
50   GOTO 20
60   DATA
70   END
```

2. Write a program that lists the names of your friends.

Flow chart
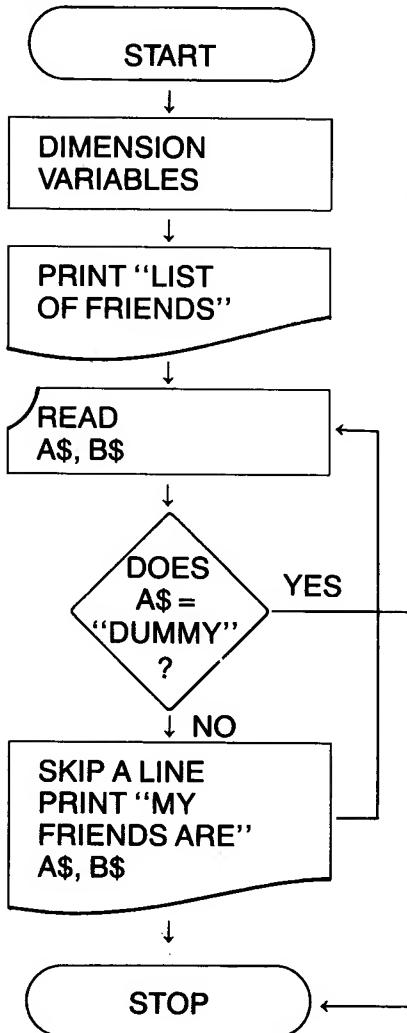
Program

```
 5 DIM A$(20),B$(20)
10  ? "LIST OF FRIENDS"
20  READ A$, B$
30  IF A$= "DUMMY" THEN 70
40  ? : ? "MY FRIENDS ARE∅";A$, B$
50  GOTO 20
60  DATA
70  END
```

CAUTION! If you use quotation marks around the DATA in line 60, the dummy data will not be read properly, and line 30 will not work.

PROGRAMMER'S PASTIME #53

Answers

Use the problem-solving approach to get ATARI to solve the following problems.

Problem 1

The teacher gave your class a test on programming the computer. The test scores were:

Jill Jarvis	73%	Your teacher needs to
Katie O'Keefe	98%	know the **average** test
Tommy Temple	67%	score.
Susie Sunbeam	82%	
You	90%	

Write a program that tells ATARI to calculate and print the average score.

HINT: To find the average of 5 numbers, add them together and divide by 5.

1. THINK about the problem.
2. Make your DATA TABLE here.

Input variables	Program variable
A = 73	T = Total
B = 98	(A+B+C+D+E)
C = 67	
D = 82	
E = 90	

3. Write the ALGORITHM (steps and equations).
   1. add up the scores
   2. divide the total by 5

4. Flow Chart

```
        ╭─────────────╮
        │    START    │
        ╰─────────────╯
               ↓
        ┌─────────────────┐
        │  CLEAR SCREEN   │
        └─────────────────┘
               ↓
        ┌─────────────────┐
        │ ? FINDING THE   │
        │ AVERAGE OF 5    │
        │ TEST SCORES     │
        └─────────────────┘
               ↓
        ┌─────────────────┐
       /│ READ SCORES     │
        │ A, B, C, D, E   │
        └─────────────────┘
               ↓
        ┌─────────────────────┐
        │ ? "JARVIS", A "%"   │
        │ "O'KEEFE", B "%"    │
        │ "TEMPLE", C "%"     │
        │ "SUNBEAM", D "%"    │
        │ "ME", E "%"         │
        └─────────────────────┘
               ↓
        ┌─────────────────┐
        │ LET T = A+B+    │
        │ C+D+E           │
        └─────────────────┘
               ↓
        ┌─────────────────┐
        │ ? "AVERAGE      │
        │ IS" T/5 "%"     │
        └─────────────────┘
               ↓
        ╭─────────────╮
        │    STOP     │
        ╰─────────────╯
```
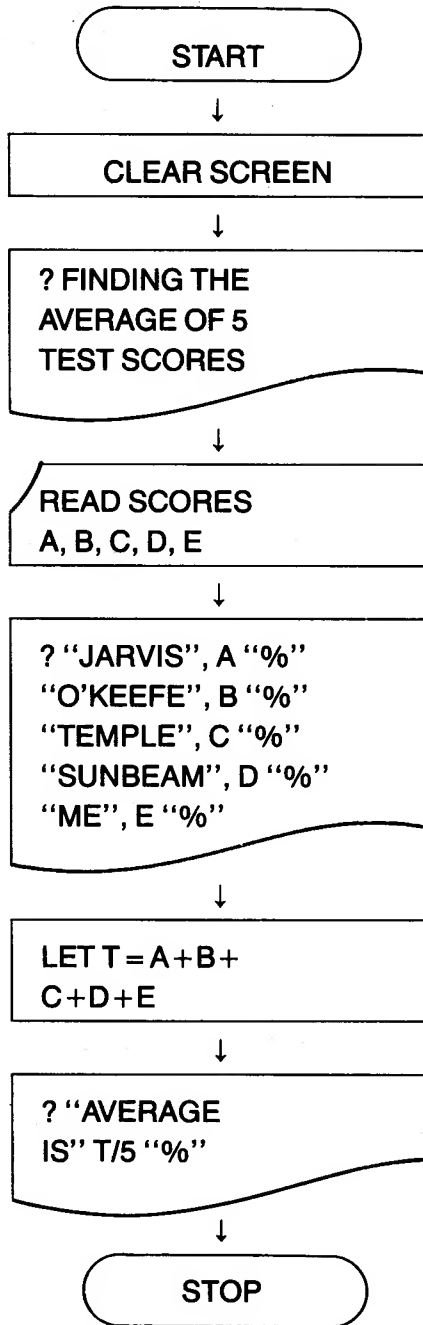
5. CODE the program

```
 10 ? " [ESC] [SHIFT] [CLEAR <] "
 20 ? "FINDING THE AVERAGE"
 25 ? "OF 5 TEST SCORES"
 30 ? : ?
 40 READ A, B, C, D, E
 50 ? "JARVIS",A; "%"
 60 ? "O'KEEFE",B; "%"
 70 ? "TEMPLE",C; "%"
 80 ? "SUNBEAM",D; "%"
 90 ? "ME",E; "%"
100 LET T = A+B+C+D+E
110 ? : ? "AVERAGE IS",T/5 "%"
120 DATA 73, 98, 67, 82, 90
130 END
```

6. DEBUG
7. REVISE

Problem 2

You are the new manager of the "Peppy Pizza" restaurant and you need the help of a computer. Write a program that will allow you to INPUT the number of small, medium, and large pizzas sold during a day.

Have ATARI print out the total number of pizzas sold and how much money you made.

PRICES: small $4.30
          medium $5.50
          large $7.25

OUTPUT HINT:
HOW MANY PIZZAS: (SMALL, MEDIUM, LARGE)
     ?_____, _____, _____
    THERE WERE _____ PIZZAS SOLD TODAY.
      "PEPPY PIZZA" MADE $_____ .

1. THINK about the problem.
2. DATA TABLE

Input variables
A = small pizzas sold
B = medium pizzas sold
C = large pizzas sold
S = 4.30 = price of small pizza
M = 5.50 = price of medium pizza
L = 7.25 = price of large pizza

Program variables
ZT = # of small pizzas * 4.30
MT = # of medium pizzas * 5.50
LT = # of large pizzas * 7.25

Output variables
T = total number of pizzas sold
TT = total amount of money made

3. ALGORITHM
   1. INPUT number of each type of pizza sold.
   2. Add number of pizzas together to find out how many were sold.
   3. Number of small pizzas * 4.30 = money made
      Number of medium pizzas * 5.50 = money made
      Number of large pizzas * 7.25 = money made
   4. Add up all of the money that was made to find the total.

4. Flow Chart

```
         ( START )
             ↓
     [ CLEAR SCREEN ]
             ↓
   ? "PEPPY PIZZA
   DAILY ACCOUNT"
   :?:?
             ↓
   ? "HOW MANY
   SMALL PIZZAS
   WERE SOLD TODAY"
             ↓
   / INPUT A
             ↓
   ? "HOW MANY
   MEDIUM PIZZAS
   WERE SOLD TODAY";
             ↓
   / INPUT B
             ↓
   ? "HOW MANY
   LARGE PIZZAS
   WERE SOLD TODAY";
             ↓
   / INPUT C
             ↓
   LET T = A+B+C
   LET S = 4.30        →   LET ZT = A*S
   LET M = 5.50            LET MT = B*M
   LET L = 7.25           LET LT = C*L
                           LET TT = ST+MT+LT
                                    ↓
                           ?:?:
                           ? "THERE WERE";
                           T;PIZZAS SOLD TODAY"
                                    ↓
                           ?:? "PEPPY PIZZA
                           MADE $";TT
                                    ↓
                                ( STOP )
```
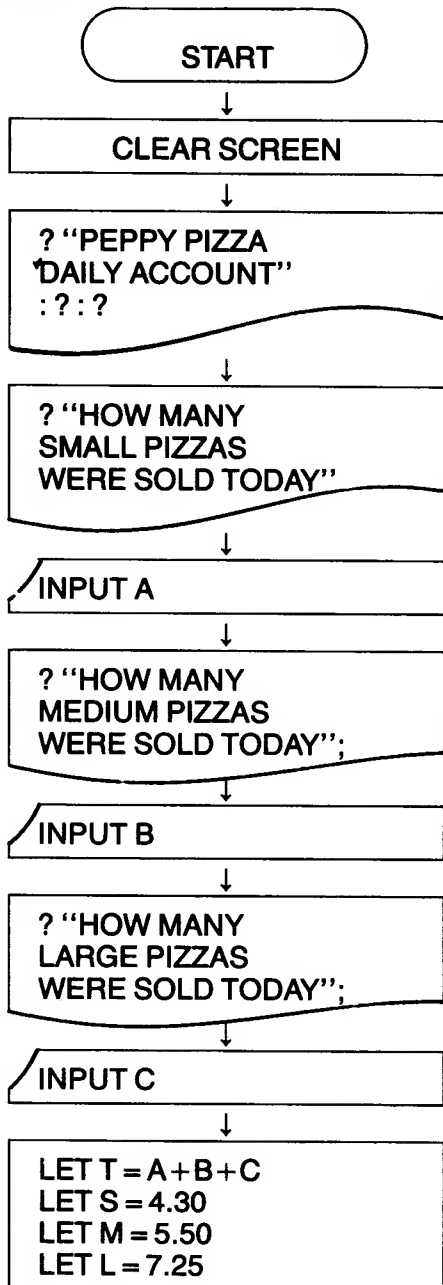
5. CODE

10 ? "[ESC] [SHIFT] [CLEAR <] "
20 ? "PEPPY PIZZA DAILY ACCOUNT"
    :?:?
30 ? "HOW MANY SMALL PIZZAS
    WERE"
40 ? "SOLD TODAY?";
50 INPUT A
60 ? "HOW MANY MEDIUM PIZZAS
    WERE"
65 ? "SOLD TODAY?";
70 INPUT B
80 ? "HOW MANY LARGE PIZZAS
    WERE"
85 ? "SOLD TODAY?";
90 INPUT C
100 LET T=A+B+C
110 LET S=4.30: LET M=5.50: LET
    L=7.25
120 LET ZT=A*S: LET MT=B*M:
    LET LT=C*L
130 LET TT=ST+MT+LT
140 ?:?:? "THERE WEREᴥ";T:
    "ᴥPIZZAS SOLD TODAY"
150 ?:?:? "PEPPY PIZZA MADE $";TT
160 END

6. DEBUG
7. REVISE

Ideas to shorten the program
30 ? "HOW MANY PIZZAS WERE SOLD"
40 ? "TODAY—SMALL, MEDIUM, LARGE";
50 INPUT A, B, C

Omit lines 60–90 and 110
120 LET ZT=A*4.30: LET MT=B*5.50
125 LET LT=C*7.25

168
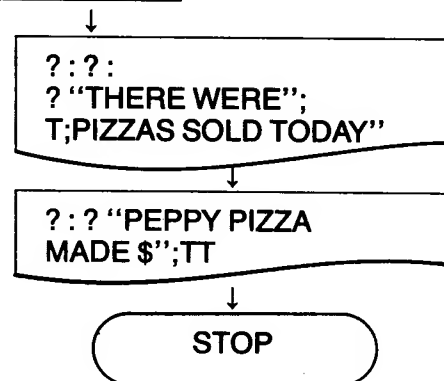
Problem 3

Write a program that will allow you to INPUT your age in years, months, and days. Example: 9 years, 3 months, 17 days.

Have ATARI calculate and print how many days, hours, and minutes old you are.

HINT: There are normally 365 days in a year and 30 days in a month. There are exactly 24 hours in a day and 60 minutes in an hour.

1. THINK about the problem.
2. DATA TABLE

Input variables	Output variables
Y=years in age	TD=total days old
M=months since your birthday	TH=total hours old
E=days since your birthday	TM=total minutes old

3. ALGORITHM
    1. Input years old, months since birthday, days since birthdate
    2. Calculate days old : Y*365+M*30+D
    3. Calculate hours old : days old*24
    4. Calculate minutes old : hours old*60

4. Flow Chart

```
        ╭─────────────────╮
        │      START      │
        ╰─────────────────╯
                 │
                 ▼
   ┌─────────────────────────┐
   │ CLEAR SCREEN            │
   │ ? "I'LL TELL YOU        │
   │ EXACTLY                 │
   │ HOW OLD YOU ARE":       │
   │ ? "HOW MANY YEARS       │
   │ OLD ARE YOU"            │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ INPUT Y                 │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ ? "HOW MANY             │
   │ MONTHS SINCE IT WAS     │
   │ YOUR BIRTHDAY"          │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ INPUT M                 │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ ? "TYPE THE ANSWER      │
   │ TO THIS EQUATION" : ? : │
   │ ? "TODAY'S DATE—        │
   │ YOUR BIRTHDATE = "      │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ INPUT D                 │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ LET TD = Y*365 +        │
   │ M*30+D : ?              │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ ? "YOU ARE";TD;         │
   │ "DAYS OLD"              │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ LET TH = TD*24          │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ ? : ? "YOU ARE";        │
   │ TH;"HOURS OLD"          │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐
   │ LET TM = TH*60          │
   └─────────────────────────┘
                 │
                 ▼
   ┌─────────────────────────┐        ╭─────────────────╮
   │ ? : ? "YOU ARE";        │   →    │      STOP       │
   │ TM;"MINUTES OLD"        │        ╰─────────────────╯
   │ ? : ? "AMAZING!"        │
   └─────────────────────────┘
```
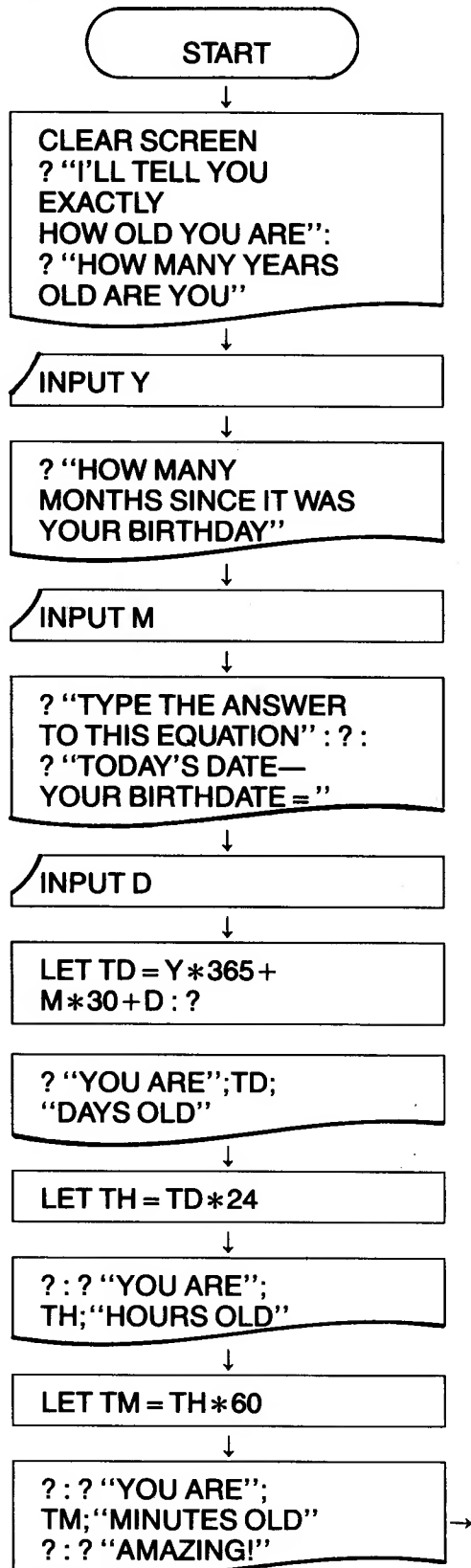
5. CODE

```
10  ? " [ESC] [SHIFT] [CLEAR <] "
20  ? "I'LL TELL YOU EXACTLY HOW
    OLD YOU ARE"
30  ? : ? "HOW MANY YEARS OLD ARE
    YOU?";
40  INPUT Y
50  ? "HOW MANY MONTHS SINCE IT
    WAS YOUR"
55  ? "BIRTHDAY?";
60  INPUT M
70  ? "TYPE THE ANSWER TO THIS
    EQUATION"
80  ? : ? "TODAY'S DATE – YOUR"
90  ? "BIRTHDATE = ";
100 INPUT D
110 LET TD = Y*365+M*30+D
120 ? : ? "YOU ARE∅";TD;"∅DAYS
    OLD"
130 LET TH = TD*24
140 ? : ? "YOU ARE∅";TH;"∅HOURS
    OLD"
150 LET TM = TH*60
160 ? : ? "YOU ARE∅";TM;"∅MINUTES
    OLD"
170 ? : ? "AMAZING!"
180 END
```

6. DEBUG
7. REVISE

Problem 4

You just got hired as a SUPER SCOOPER at the DIPPER DELIGHT Ice Cream Store. Write a program that will allow you to INPUT how many hours you worked for the week.

Have ATARI calculate and print hours worked and your salary for the week if you make $3.25 an hour.

OUTPUT HINT:

HOW MANY HOURS DID YOU WORK? _____

    YOU WORKED _____ HOURS AND MADE

                $_____.

1. THINK about the problem.
2. DATA TABLE
   Input variables
   H=hours worked for the week
   S=salary=$3.25 hour

3. ALGORITHM
   1. Input hours worked for the week.
   2. Multiply number of hours by 3.25

4. Flow Chart
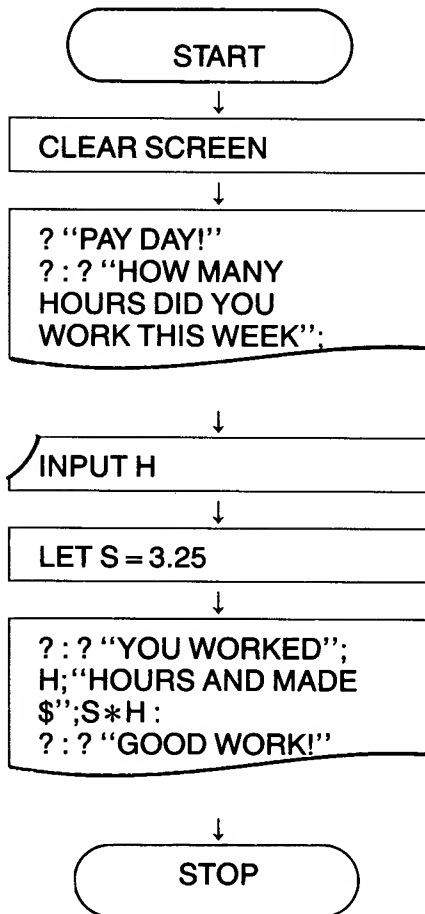
```
        ( START )
            ↓
   [ CLEAR SCREEN ]
            ↓
   ┌──────────────────┐
   │ ? "PAY DAY!"     │
   │ ? : ? "HOW MANY  │
   │ HOURS DID YOU    │
   │ WORK THIS WEEK"; │
   └──────────────────┘
            ↓
   / INPUT H
            ↓
   [ LET S = 3.25 ]
            ↓
   ┌──────────────────┐
   │ ? : ? "YOU WORKED"; │
   │ H;"HOURS AND MADE   │
   │ $";S*H :             │
   │ ? : ? "GOOD WORK!"   │
   └──────────────────┘
            ↓
        ( STOP )
```

5. CODE

```
10 ? " [ESC] [SHIFT] [CLEAR<] "
20 ? "PAY DAY!"
30 ? "HOW MANY HOURS DID"
35 ? "YOU WORK THIS WEEK";
40 INPUT H
50 LET S=3.25
60 ? : ? "YOU WORKEDØ";H;
   "ØHOURS"
70 ? "AND MADE $";S*H
80 ? : ? "GOOD WORK!"
90 END
```

6. DEBUG
7. REVISE

Problem 5

Add to the problem you wrote for PROBLEM 4 so that ATARI can calculate overtime pay. (Overtime is any hours worked **over** 40 hours a week.) You get paid $4.75 for every hour of overtime you work.

Add this to our OUTPUT:

YOU WORKED _____ OVERTIME HOURS AND
MADE $_____ IN OVERTIME.
YOUR TOTAL PAY FOR THE WEEK IS $_____.
(Total pay is regular pay + overtime pay.)

HINT: You will need a decision box in your flow chart to ask:

IS H > 40?

1. THINK about the problem.
2. DATA TABLE

Input variables	**Output variables**
H = hours worked for the week	OH = overtime hours worked
S = salary = $3.25 hour	RH = regular hours worked
OP = overtime pay = $4.75 hour	

3. ALGORITHM
   1. Input hours worked for the week.
   2. Calculate how many hours overtime you worked.
   3. Multiply regular hours by 3.25
   4. Multiply overtime hours by 4.75.
   5. Calculate total salary.

4. Flow Chart
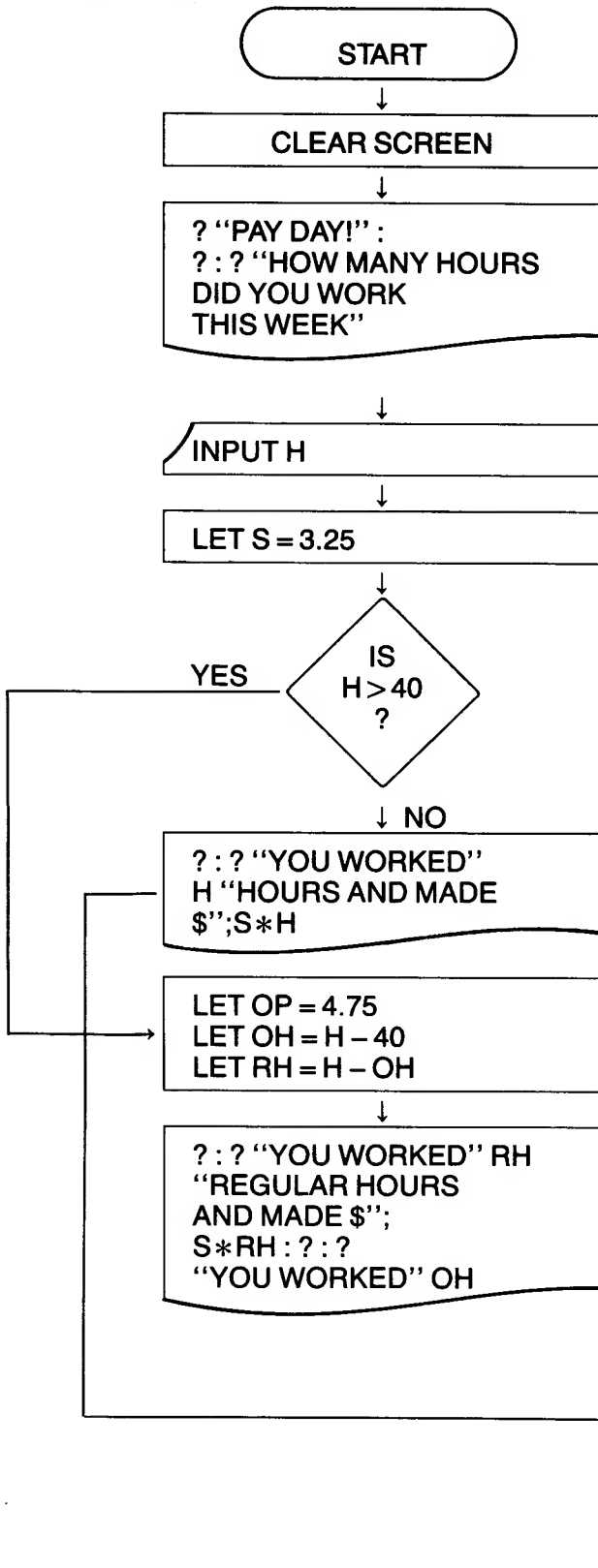
```
          ┌─────────────┐
          │    START    │
          └──────┬──────┘
                 ↓
          ┌─────────────┐
          │ CLEAR SCREEN│
          └──────┬──────┘
                 ↓
    ┌────────────────────────┐
    │ ? "PAY DAY!" :         │
    │ ? : ? "HOW MANY HOURS  │
    │ DID YOU WORK           │
    │ THIS WEEK"             │
    └────────────┬───────────┘
                 ↓
    ┌────────────────────────┐
    │ INPUT H                │
    └────────────┬───────────┘
                 ↓
    ┌────────────────────────┐
    │ LET S = 3.25           │
    └────────────┬───────────┘
                 ↓
              ◇ IS
       YES    H > 40 ?
                 │ NO
                 ↓
    ┌────────────────────────┐
    │ ? : ? "YOU WORKED"     │
    │ H "HOURS AND MADE      │
    │ $";S∗H                 │
    └────────────┬───────────┘
                 ↓
    ┌────────────────────────┐
    │ LET OP = 4.75          │
    │ LET OH = H – 40        │
    │ LET RH = H – OH        │
    └────────────┬───────────┘
                 ↓
    ┌────────────────────────┐      ┌────────────────────────┐
    │ ? : ? "YOU WORKED" RH  │      │ ? "OVERTIME HOURS      │
    │ "REGULAR HOURS         │  →   │ AND MADE $" OH∗OP       │
    │ AND MADE $";           │      │ "IN OVERTIME" ?:?:      │
    │ S∗RH : ? : ?           │      │ "YOUR TOTAL SALARY      │
    │ "YOU WORKED" OH         │      │ IS $";OH∗OP+S∗RH        │
    └────────────────────────┘      └────────────┬───────────┘
                                                 ↓
                          ┌────────────────────────┐
                          │ ? : ? "GOOD            │
                          │ WORK!"                 │
                          └────────────┬───────────┘
                                       ↓
                                ┌─────────────┐
                                │    STOP     │
                                └─────────────┘
```

5. CODE

```
 10 ? " [ESC] [SHIFT] [CLEAR/<] "
 20 ? "PAY DAY!"
 30 ? : ? "HOW MANY HOURS DID"
 35 ? "YOU WORK THIS WEEK";
 40 INPUT H
 50 LET S=3.25
 60 IF H>40 THEN 90
 70 ? : ? "YOU WORKEDb";H; "bHOURS"
 75 ? "AND MADE $";S∗H
 80 GOTO 180
 90 LET OP=4.75
100 LET OH=H–40
110 LET RH=H–OH
120 ? : ? "YOU WORKEDb";RH; "bREGULAR HOURS"
130 ? "AND MADE $";S∗RH
140 ? : ? "YOU WORKEDb";OH; "bOVERTIME"
150 ? "HOURS AND MADE $";OH∗OP
160 ? "IN OVERTIME"
165 ? : ? "YOUR TOTAL SALARY ISb";
170 ? "$";OH∗OP+S∗RH
180 ? : ? "GOOD WORK!"
190 END
```
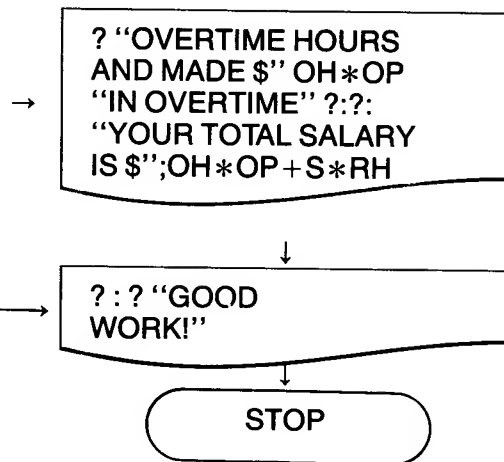
6. DEBUG
7. REVISE

Problem 6

You are the famous sportscaster H.E. Nosell. You have been asked to calculate the batting averages of Big League Baseball players. Write a program that allows you to INPUT a player's name, hits, and times at bat.

Have ATARI calculate and print the player's name and batting average.

HINT: To calculate batting average, use this equation:

$$1000*hits/times\ at\ bat$$

1. THINK about the problem.
2. DATA TABLE

Input variables

N$=Player's
    name
H=Hits made
T=Times up to bat

Output variable

A=Batting
Average=$1000*H/T$

3. ALGORITHM
   1. Input name, hits, times at bat
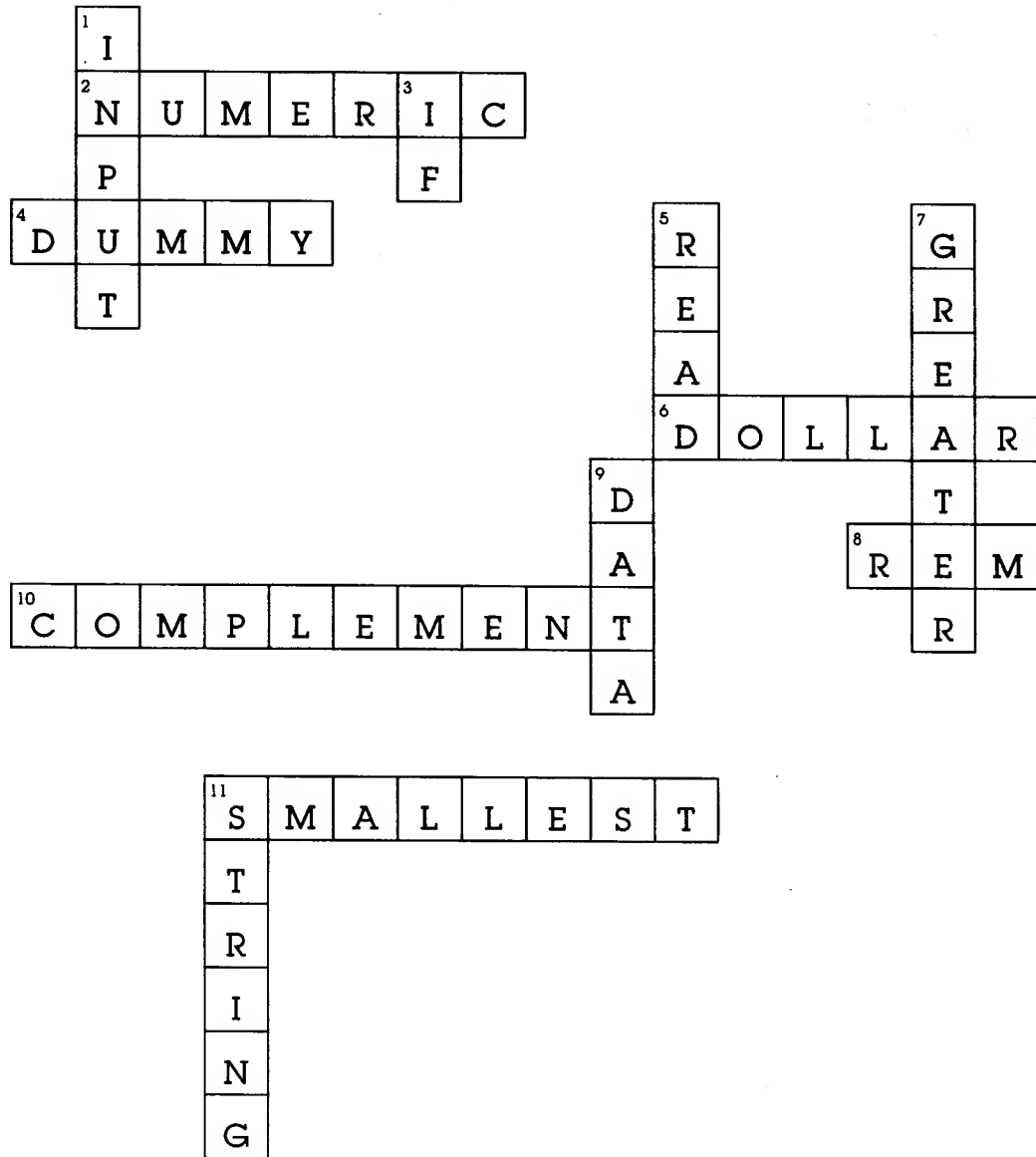   2. Calculate batting average: $1000*H/T$

4. Flow Chart

```
        ┌─────────────────┐
        │      START      │
        └─────────────────┘
                 ↓
   ┌───────────────────────────┐
   │   DIMENSION VARIABLE      │
   └───────────────────────────┘
                 ↓
   ┌───────────────────────────┐
   │  ? "TYPE PLAYER'S         │
   │  NAME, HITS,              │
   │  AND TIMES AT BAT"        │
   └───────────────────────────┘

                 ↓
   ┌───────────────────────────┐
   │  INPUT N$, H, T           │
   └───────────────────────────┘
                 ↓
   ┌───────────────────────────┐
   │  LET A = 1000 * H/T       │
   └───────────────────────────┘
                 ↓
   ┌───────────────────────────┐
   │  ? : ? "BATTING AVERAGE   │
   │  OF";N$;"IS";A            │
   └───────────────────────────┘

                 ↓
        ┌─────────────────┐
        │      STOP       │
        └─────────────────┘
```
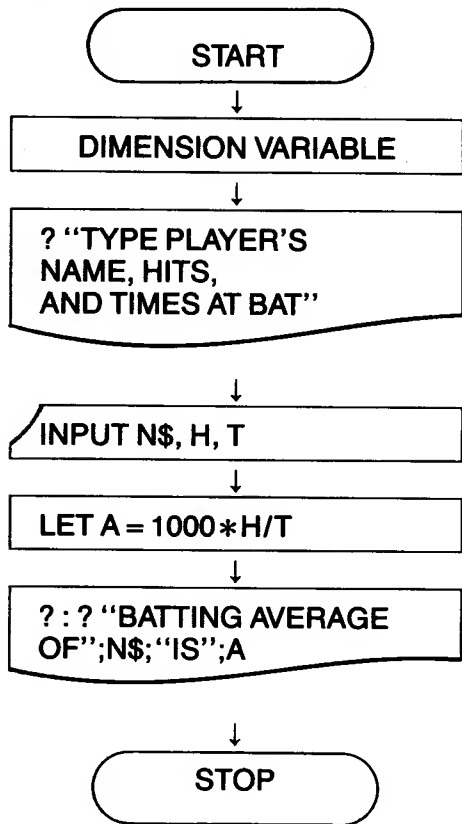
5. CODE

```
10 ? DIM N$(25)
20 ? "TYPE PLAYER'S NAME, HITS"
30 ? "AND TIMES AT BAT"
40 INPUT N$, H, T
50 LET A=1000*H/T
60 ? : ? "BATTING AVERAGE OF∅";N$
70 ? "IS∅";A
80 END
```

6. DEBUG
7. REVISE

COMPONENT 6 FUN PAGE

Answers

```
      1
      I
   2              3
   N U M E R I C
   P           F
4
D U M M Y
   T
```

```
         5           7
         R           G
         E           R
         A           E
      6              8
      D O L L A R
   9              T
   D           8
10                R E M
C O M P L E M E N T A
   A           R
```

```
11
S M A L L E S T
T
R
I
N
G
```

PROGRAMMER'S PASTIME #54

Answers

For each conversion problem, identify the important parts by writing: HEADING, FOR–NEXT LOOP, CONVERSION EQUATION next to the lines in the program. Then write what you think ATARI would print as the output.

Program Example:	Important parts	Output	
10 REM CONVERT FEET TO METERS		FEET	METERS
		1	0.3
20 ? "FEET", "METERS"	HEADING	2	0.6
30 ?		3	0.9
40 FOR F=1 TO 10	FOR–NEXT LOOP	4	1.2
50 ? F, F*.3	CONVERSION	5	1.5
60 NEXT F	EQUATION	6	1.8
70 END		7	2.1
		8	2.4
		9	2.7
		10	3

	Program	Important parts	Output	
1.	10 REM CONVERT FEET TO YARDS		FEET	YARDS
			1	.333333333
	20 ? "FEET", "YARDS"	HEADING	3	1
			5	1.66666666
	30 ?		7	2.33333333
	40 FOR F=1 TO 12 STEP 2	FOR–NEXT LOOP	9	3
			11	3.66666666
	50 ? F, F/3	CONVERSION		
	60 NEXT F	EQUATION		
	70 END			

Program	Important parts	Output	

2.
```
10 REM CONVERT
   TEASPOONS TO
   TABLESPOONS
20 ? "TEASPOONS",
   "TABLESPOONS"
30 FOR T = 3 TO 18
   STEP 3
40 ? T, T/3
50 NEXT T
60 END
```

Important parts:
HEADING

FOR–NEXT LOOP

CONVERSION
   EQUATION

Output:

TEASPOONS	TBSP
3	1
6	2
9	3
12	4
15	5
18	6

3.
```
10 REM CONVERT
   POUNDS TO OUNCES
20 ? "POUNDS",
   "OUNCES"
30 FOR P = 1 TO 6
40 ? P, P * 16
50 NEXT P
60 END
```

Important parts:
HEADING

FOR–NEXT LOOP
CONVERSION
   EQUATION

Output:

POUNDS	OUNCES
1	16
2	32
3	48
4	64
5	80
6	96

4.
```
10 REM CONVERT
   YARDS TO INCHES
20 ? "YARDS",
   "INCHES"
30 FOR Y = 1 TO 5
40 ? Y, Y * 36
50 NEXT Y
60 END
```

Important parts:
HEADING

FOR–NEXT LOOP
CONVERSION
   EQUATION

Output:

YARDS	INCHES
1	36
2	72
3	108
4	144
5	180

PROGRAMMER'S PASTIME #55

Write a conversion program for each problem. Make sure your program has a heading, FOR-NEXT loop, and conversion equation. Run your programs on ATARI to check for bugs.

Problem

1. Convert 1-20 inches to centimeters.
   CONVERSION EQUATION:
   Centimeters=I*2.5

2. Convert 1-20 kilometers to miles.
   CONVERSION EQUATION:
   Miles=K/1.6

3. Convert 1-20 pounds to grams.
   CONVERSION EQUATION:
   Grams=P*454

Program

```
10 REM CONVERT INCHES TO
   CENTIMETERS
20 ? "INCHES" , "CENTIMETERS"
30 FOR I= 1 TO 20
40 ?I, I*2.5
50 NEXT I
60 END
```

```
10 REM CONVERT KILOMETERS TO
   MILES
20 ? "KILOMETERS" , "MILES"
30 FOR K= 1 TO 20
40 ?K, K/1.6
50 NEXT K
60 END
```

```
10 REM CONVERT POUNDS TO GRAMS
20 ? "POUNDS" , "GRAMS"
30 FOR P= 1 TO 20
40 ?P, P*454
50 NEXT P
60 END
```

Problem	Program
4. Convert 1–10 liters to quarts. CONVERSION EQUATION: QUARTS=L/3.8	10 REM CONVERT LITERS TO QUARTS 20 ? "LITERS" , "QUARTS" 30 FOR L=1 TO 10 40 ? L, L/3.8 50 NEXT L 60 END
5. Convert 0°–100° Fahrenheit to Celsius. CONVERSION EQUATION: °C=5*(F−32)/9	10 REM CONVERT °C to °F 20 ? "FAHRENHEIT" , "CELSIUS" 30 FOR F=0 TO 100 40 ? F,, 5*(F−32)/9 50 NEXT F 60 END
6. Convert 1–100 pounds to kilograms. CONVERSION EQUATION: Kilograms=P*.45	10 REM CONVERT POUNDS TO KILOGRAMS 20 ? "POUNDS" , "KILOGRAMS" 30 FOR P=1 TO 100 40 ? P, P*.45 50 NEXT P 60 END

PROGRAMMER'S PASTIME #56

Answers

Use the problem-solving approach to get ATARI to solve the following conversion problems.

A. Jed needs to find out what decimal $6/7$ stands for. Write a program that lists the fractions $1/7$ through $7/7$ and the decimals they stand for. CONVERSION EQUATION: Decimal = X/7

1. THINK about the problem.
2. DATA TABLE
   Program variable
   F = numerator = 1 to 7

3. ALGORITHM
   1. FOR–NEXT LOOP should loop from 1 to 7.
   2. Conversion equation is F/7.

4. Flow Chart

```
        ┌─────────────────────┐
        (        START        )
        └─────────────────────┘
                  ↓
        ┌─────────────────────┐
        │ CLEAR SCREEN        │
        │ ? "FRACTIONS",      │
        │ "DECIMALS" : ?      │
        └─────────────────────┘
                  ↓
        ┌─────────────────────┐
        │ FOR F = 1 TO 7      │
        └─────────────────────┘
                  ↓
        ┌─────────────────────┐
        │ ? F "/7" ,          │
        │ F/7                 │
        └─────────────────────┘
                  ↓
        ┌─────────────────────┐
        │ NEXT F              │
        └─────────────────────┘
                  ↓
        ┌─────────────────────┐
        (        STOP         )
        └─────────────────────┘
```
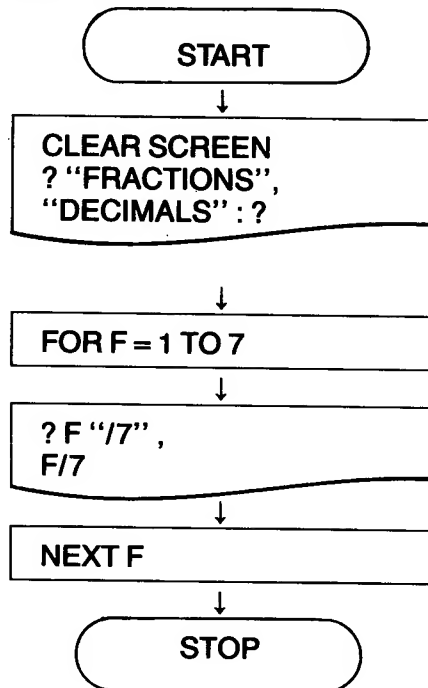
5. CODE

```
10 ? " [ESC] [SHIFT] [CLEAR<] "
20 ? "FRACTIONS" , "DECIMALS"
30 ?
40 FOR F = 1 TO 7
50 ? F; "/7" , F/7
60 NEXT F
70 END
```

6. DEBUG
7. REVISE

B. Amy Astronaut is going to the moon. She learned that because the gravity on the moon is only ⅙ of the earth's gravity, she will weigh less on the moon. Write a program that asks you to INPUT how much you weigh. Then have ATARI print how much you would weigh on the moon. CONVERSION EQUATION: moon weight=earth weight/6

1. THINK about the problem
2. DATA TABLE
   Input variable
   W=your weight on earth

3. ALGORITHM
   1. Input weight on earth
   2. Calculate moon weight: earthweight/6

4. Flow Chart

```
        ┌─────────────────────┐
        (       START         )
        └─────────────────────┘
                   ↓
    ┌──────────────────────────┐
    │ ? CLEAR SCREEN           │
    │ ? "HOW MUCH              │
    │ DO YOU WEIGH"            │
    └──────────────────────────┘
                   ↓
    ┌──────────────────────────┐
    / INPUT W                  │
    └──────────────────────────┘
                   ↓
    ┌──────────────────────────┐
    │ ? : ? "ON THE            │
    │ MOON YOU WOULD           │
    │ WEIGH";W/6;"POUNDS"      │
    └──────────────────────────┘
                   ↓
        ┌─────────────────────┐
        (       STOP          )
        └─────────────────────┘
```
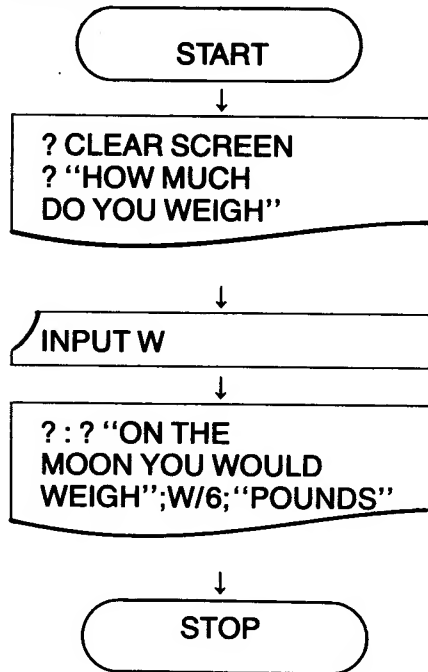
5. CODE

```
10 ? " [ESC] [SHIFT] [CLEAR/<] "
20 ? "HOW MUCH DO YOU WEIGH";
30 INPUT W
40 ? : ? "ON THE MOON YOU WOULD"
50 ? "WEIGHø";W/6;"øPOUNDS"
60 END
```

6. DEBUG
7. REVISE

C. Add to program #2 so ATARI will print a conversion table of weight on earth from 10 pounds to 100 pounds and the equal moon weights **after** printing the output in program #2.

1. Flow Chart

.
.
.

```
+---------------------------+
| ? : ? "EARTH              |
| WEIGHT",                  |
| "MOON                     |
| WEIGHT" : ?               |
+---------------------------+
            ↓
+---------------------------+
| FOR W = 10                |
| TO 100                    |
+---------------------------+
            ↓
+---------------------------+
| ? W, , W/6                |
+---------------------------+
            ↓
+---------------------------+
| NEXT W                    |
+---------------------------+
            ↓
(        STOP        )
```
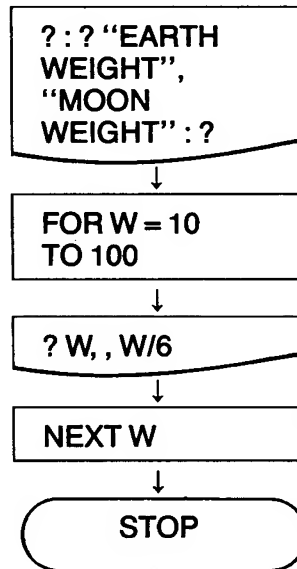
2. CODE

```
60 ? : ? "EARTH
      WEIGHT" , "MOON
      WEIGHT"
70 ?
80 FOR W = 10 TO 100
90 ? W , , W/6
100 NEXT W
120 END
```

3. DEBUG
4. REVISE

CHALLENGE

D. Fred's class took a test in which there were 20 questions asked. Fred's score was 16 correct out of 20, or $\frac{16}{20}$. Fred wants to know what percentage this would be. Write a program that lists the percentages for the test scores $\frac{1}{20}$ through $\frac{20}{20}$.

$$\frac{X}{20} = \begin{array}{l}\text{number answered correctly} \\ \text{total number of questions}\end{array}$$

$$\frac{P}{100} = \begin{array}{l}\text{percentage} \\ \text{total}\end{array}$$

CONVERSION EQUATION: $P = X * 100/20$

1. THINK about the problem
2. DATA TABLE
   Program variable
   S = scores = 1 to 20

3. ALGORITHM
   1. FOR–NEXT LOOP should loop from 1 to 20.
   2. Conversion equation is $S * 100/20$.

4. Flow Chart

```
          ┌─────────────────┐
          (     START       )
          └─────────────────┘
                  ↓
     ┌──────────────────────┐
     │ ? CLEAR SCREEN       │
     │ ? "SCORES",          │
     │ "%" : ?              │
     └──────────────────────┘
                  ↓
     ┌──────────────────────┐
     │ FOR S = 1 TO 20      │
     └──────────────────────┘
                  ↓
     ┌──────────────────────┐
     │ ? S;"/20",           │
     │ S*100/20             │
     └──────────────────────┘
                  ↓
     ┌──────────────────────┐
     │ NEXT S               │
     └──────────────────────┘
                  ↓
          ┌─────────────────┐
          (      END        )
          └─────────────────┘
```
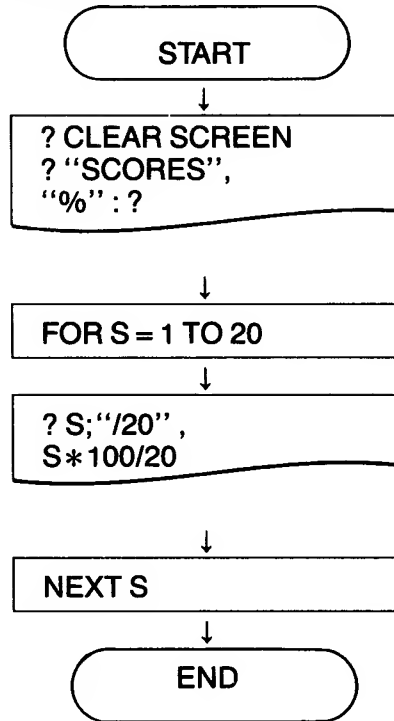
5. CODE

```
10 ? " [ESC] [SHIFT] [CLEAR <] "
20 ? "SCORES" , "%"
30 ?
40 FOR S = 1 TO 20
50 ? S;"/20" , S*100/20
60 NEXT S
70 END
```

6. DEBUG
7. REVISE

CHALLENGE

E. Change program #4 so ATARI asks you to INPUT how many test questions there were (T), and how many questions you answered correctly (C). Have ATARI print your score and the percentage you got correct.

HINT: score = C out of T

percentage = C * 100 / T

1. Flow Chart

```
        ( START )
           ↓
  ? CLEAR SCREEN
  ? "FINDING OUT
  YOUR SCORE" : ? :
  ? "HOW MANY TEST
  QUESTIONS"
           ↓
  /INPUT T
           ↓
  ? : ? "HOW MANY
  DID YOU ANSWER
  CORRECTLY"
           ↓
  /INPUT C
           ↓
  ? : ? "YOU SCORED"
  C;"/";T : ? : ?
  "THAT IS";
  C*100/T;"%"
           ↓
        ( STOP )
```
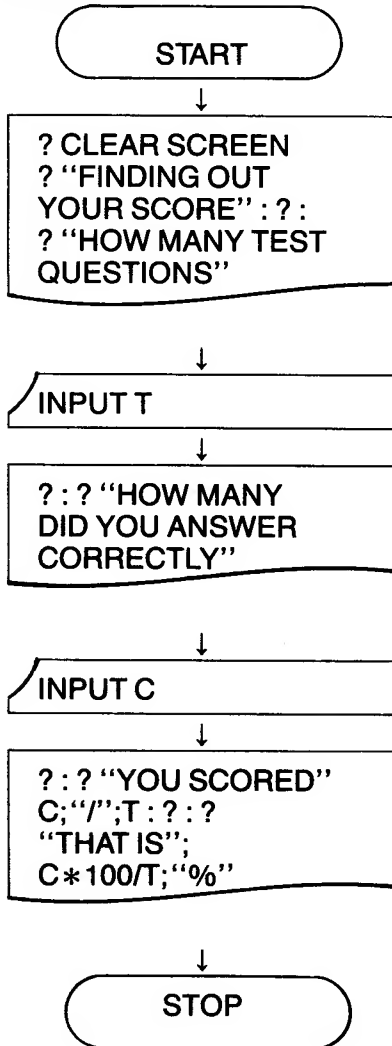
2. CODE

```
10 ? " [ESC] [SHIFT] [CLEAR/<] ''
20 ? "FINDING OUT YOUR SCORE"
30 ? : ? "HOW MANY TEST
   QUESTIONS";
40 INPUT T
50 ? : ? "HOW MANY DID YOU"
55 ? "ANSWER CORRECTLY";
60 INPUT C
70 ? : ? "YOU SCOREDø";C;"/";T
80 ? : ? "THAT ISø";C*100/T;"%"
90 END
```

3. DEBUG
4. REVISE

PROGRAMMER'S PASTIME #57

Answers

RUN the program three times on ATARI. Each time the program is run, write down the random numbers that ATARI printed. Then write the lowest and highest numbers in the list. Run the program several more times and visually note the highest and lowest numbers.

RUN #1

numbers					
lowest					
highest					

RUN #2

numbers					
lowest					
highest					

RUN #3

numbers					
lowest					
highest					

Program

```
10 FOR L=1 TO 5
20  LET X=10*RND(1)
30  ? X
40 NEXT L
50 END
```

Answers will vary.

PROGRAMMER'S PASTIME #58

Answers

Read each RND function. Figure out what the lowest and highest random numbers will be that ATARI could print.

Function	ATARI will print random numbers between and including:		
Example:			
LET X=18*RND(1)	0	and	17.9999
1. LET X=300*RND(1)	0	and	299.9999
2. LET X=RND(1)	0	and	.9999
3. LET X=3*RND(1)	0	and	2.9999
4. LET X=67*RND(1)	0	and	66.9999
5. LET X=100*RND(1) +1	1	and	100.9999
6. LET X=25*RND(1)+1	1	and	25.9999
7. LET X=116*RND(1) +1	1	and	116.9999
8. LET X=39*RND(1)+1	1	and	39.9999
9. LET X=436*RND(1)	0	and	435.9999
10. LET X=77*RND(1)+1	1	and	77.9999
11. LET X=43*RND(1)+1	1	and	43.9999
12. LET X=13*RND(1)	0	and	12.9999
13. LET X=59*RND(1)+1	1	and	59.9999

PROGRAMMER'S PASTIME #59

INTEGERS are whole numbers. The INT function rounds DOWN to the next whole number to make it an integer. Read each INT function. Then write what ATARI would print for the output.

Function	Output
Example:	
10 LET C=4.96	4
20 ?INT(C)	
1. 10 LET X=66.823	66
20 ?INT(X)	
2. ?INT(4.89)	4
3. 10 LET R=992.01	992
20 ?INT(R)	
4. ?INT(63.49321)	63
5. 10 LET BD=−16.003	−17
20 ?INT(BD)	
6. 10 LET P1=43.001	43
20 ?INT(P1)	
7. ?INT(660.666)	660
8. ?INT(−33.23)	−34
9. 10 LET S=4120.7	4120
20 ?INT(S)	
10. ?INT(−999.999)	−1000

Answers

We use both the INT and RND functions to tell ATARI to print a random integer. Read each function. Then write the two numbers that ATARI must create random integers **between**.

	ATARI will print random
Function	**integers between:**

Example:

INT(40*RND(1)+26)   _26_ and _67_

DO: 40+26+1=67

	Function			
1.	INT(14*RND(1)+3)	3	and	18
2.	INT(221*RND(1)+99)	99	and	321
3.	INT(3*RND(1)+2)	2	and	6
4.	INT(22*RND(1)+16)	16	and	39
5.	INT(55*RND(1)+28)	28	and	84
6.	INT(77*RND(1)+75)	75	and	153
7.	INT(94*RND(1)+33)	33	and	128
8.	INT(101*RND(1)+66)	66	and	168
9.	INT(63*RND(1)+7)	7	and	71
10.	INT(80*RND(1)+45)	45	and	126
11.	INT(46*RND(1)+23)	23	and	70
12.	INT(39*RND(1)+19)	19	and	59

PROGRAMMER'S PASTIME #61

Write an RND function for each description.

Create random numbers between and including:	Function
Example:	
0 and 9.9999	LET X= 10*RND(1)
1. 0 and 14.9999	LET X=15*RND(1)
2. 0 and 92.9999	LET X=93*RND(1)
3. 1 and 45.9999	LET X=45*RND(1)+1
4. 0 and 70.9999	LET X=71*RND(1)
5. 1 and 26.9999	LET X=26*RND(1)+1
6. 0 and 106.9999	LET X=107*RND(1)
7. 0 and 66.9999	LET X=67*RND(1)
8. 1 and 211.9999	LET X=211*RND(1)+1
9. 1 and 31.9999	LET X=31*RND(1)+1
10. 1 and 441.9999	LET X=441*RND(1)+1
11. 0 and 89.9999	LET X=90*RND(1)
12. 1 and 53.9999	LET X=53*RND(1)+1
13. 1 and 382.9999	LET X=382*RND(1)+1
14. 0 and 554.9999	LET X=555*RND(1)

Answers

Write an INT and RND function for each description. Remember the equation:

$$INT((B-(A+1))*RND(1)+A)$$

B=largest number   A=smallest number

To print random integers between	Function
Example: 5 and 18	INT(12*RND(1)+5)

DO: INT((18−(5+1))*RND(1)+5)
    INT(12*RND(1)+5)

1. 16 and 48	INT(31*RND(1)+16)
2. 2 and 10	INT(7*RND(1)+2)
3. 10 and 100	INT(89*RND(1)+10)
4. 1 and 50	INT(48*RND(1)+1)
5. 33 and 99	INT(65*RND(1)+33)
6. 50 and 100	INT(49*RND(1)+50)
7. 75 and 100	INT(24*RND(1)+75)
8. 27 and 41	INT(13*RND(1)+27)
9. 62 and 300	INT(237*RND(1)+62)
10. 49 and 52	INT(2*RND(1)+49)

PROGRAMMER'S PASTIME #63

Answers

Make a flow chart and write a program for each problem. Debug your programs by running them on ATARI.

1. Write a program that will print 10 random decimals between 1 and 100 and then print the integer for each.

Flow Chart

START
↓
CLEAR SCREEN
↓
? "RANDOM
DECIMAL",
"INTEGER"

↓
FOR L = 1 TO 10
↓
LET X = 100 * RND(1) + 1
LET Y = INT(X)
↓
?
? X, Y
↓
NEXT L
↓
STOP
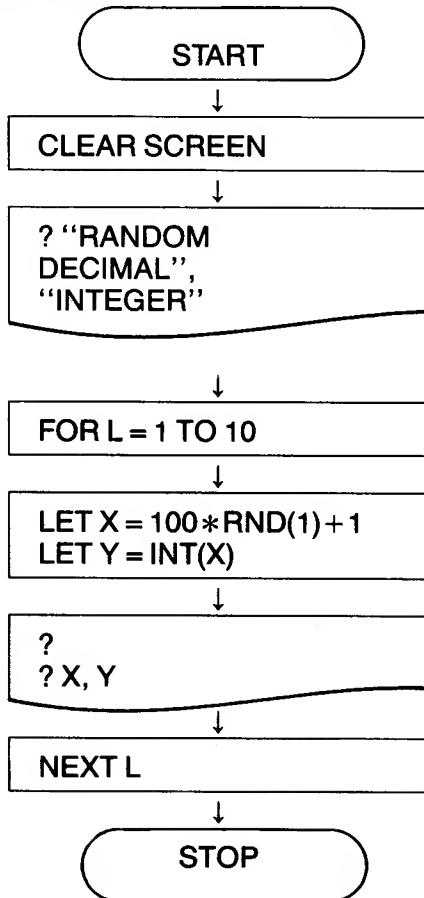
Program

```
10 ?" ESC  SHIFT  CLEAR "
              <
20 ?"RANDOM DECIMAL",
   "INTEGER"
30 FOR L=1 TO 10
40 LET X=100*RND(1)+1
50 LET Y=INT(X)
60 ?:?X,Y
70 NEXT L
80 END
```

196

2. Write a CAI program that asks a student to multiply two random numbers between 1 and 10.

Flow chart

```
        ( START )
            ↓
   [ CLEAR SCREEN ]
            ↓
   [ ? "*
      PRACTICE" ]
            ↓
   [ LET C = INT(10*
       RND(1)+1)
     LET D = INT(10*
       RND(1)+1) ]
            ↓
   [ ? : ? C;"*";D;
      "=" ; ]
            ↓
   / INPUT E /
            ↓
        < DOES
          E = C*D
          ? >  ── YES ──┐
            ↓ NO        │
   [ ? : ? "WRONG.      │
      TRY AGAIN" ]      │
            ↓           │
   [ ? : ? "THAT'S  ◄───┘
      RIGHT!" ]
```
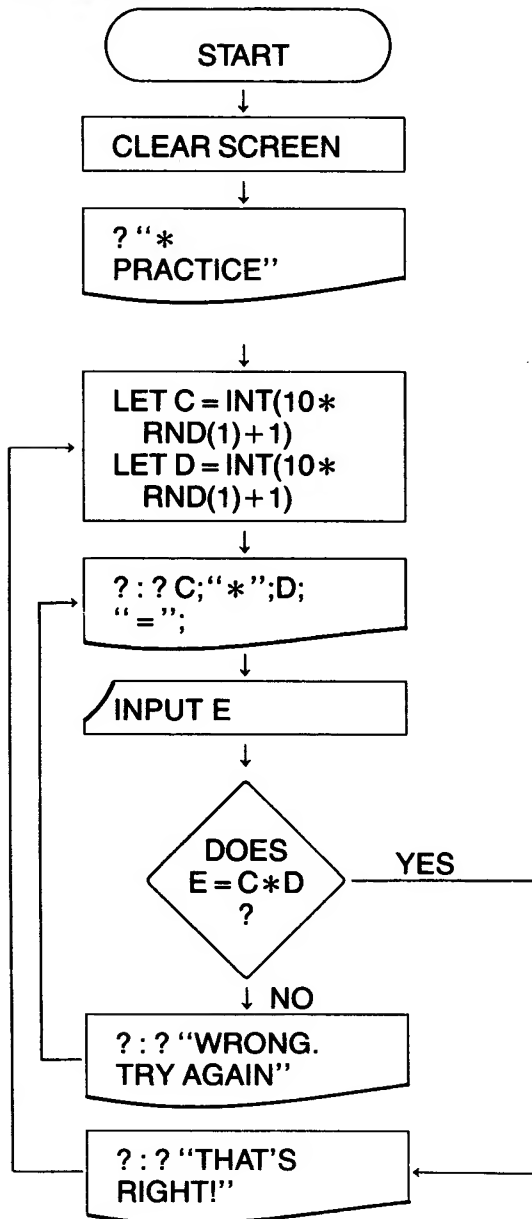
Program

```
 10 ? " [ESC] [SHIFT] [CLEAR/<] "
 20 ? "* PRACTICE"
 30 LET C=INT(10*RND(1)+)
 40 LET D=INT(10*RND(1)+1)
 50 ? : ? C;"*";D;"=";
 60 INPUT E
 70 IF E=C*D THEN 100
 80 ? "WRONG. TRY AGAIN"
 90 GOTO 50
100 ? "THAT'S RIGHT!"
120 GOTO 30
```

3. You can use the INPUT statement in a program with the RND and INT functions. Write a program so that ATARI will ask you to type in two integers. Then have ATARI print 10 random integers between those two numbers.

OUTPUT HINT:

    TYPE IN TWO NUMBERS
    AND I WILL CREATE TEN
    RANDOM INTEGERS BETWEEN
    THOSE TWO NUMBERS
    ?
    10 RANDOM INTEGERS
        BETWEEN   AND   ARE:

Flow chart

```
        ( START )
            ↓
    [ CLEAR SCREEN ]
            ↓
    ? "TYPE IN 2 NUMBERS
    AND I WILL CREATE 10
    RANDOM INTEGERS
    BETWEEN THEM"
            ↓
    / INPUT A, Z
            ↓
    ? : ? "10 RANDOM
    INTEGERS BETWEEN"
    A "AND" Z
            ↓
    [ FOR L = 1 TO 10 ]
            ↓
    LET X = INT(Z*
        RND(1)+A)
            ↓
    ? X ;
            ↓
    [ NEXT L ]
            ↓
        ( STOP )
```
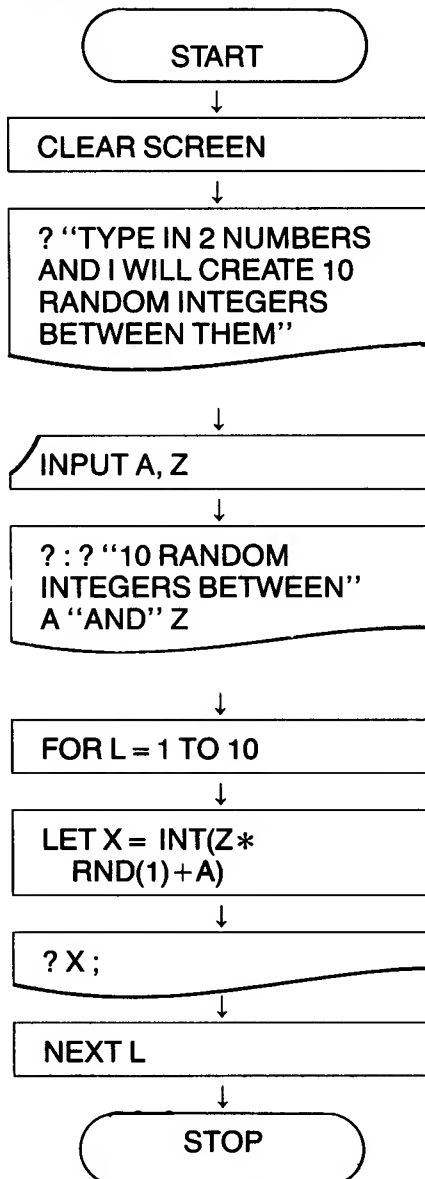
Program

```
 10 ? " [ESC] [SHIFT] [CLEAR <] "
 20 ? "TYPE IN 2 NUMBERS AND I"
 30 ? "WILL CREATE 10 RANDOM"
 40 ? "INTEGERS BETWEEN THEM"
 50 INPUT A, Z
 60 ? : ? "10 RANDOM INTEGERS"
 70 ? "BETWEEN";A;"AND";Z
 80 FOR L=1 TO 10
 90 LET X=INT(Z*RND(1)+A)
100 ? X;
110 NEXT L
120 END
```

PROGRAMMER'S PASTIME #64

1. Write your own music by adding the necessary DATA in line 60:

```
10 REM WRITE A SONG
20 READ N
30 SOUND 1, N, 10, 8
40 FOR T=1 TO 150:NEXT T
50 GOTO 20
60 DATA
```

2. The above program holds the last note until you press BREAK , type END and press RETURN . Add to the above program so that it ends by itself. (HINT—use some dummy data in line 60.)

Answers will vary

3. Rewrite the program so that Voice, then Tone, and then Loudness are altered. (Be careful when changing Loudness so that others around you are not disturbed.)

The answers will vary, but lines 20, 30, and 60 should mostly change as follows:

Voice—
20 READ X (or any variable)
30 SOUND X, 121, 10, 8 (Notes, Tone, and Loudness values may vary)
60 DATA (Data values from 0 to 3 in any order)

Tone—
20 READ X (or any variable)
30 SOUND 1, 121, X, 8 (Voice, Notes, and Loudness values may vary)
60 DATA (Data values may vary from 0 to 14, just the even numbers, in any order.)

Loudness—
20 READ X (or any variable)
30 SOUND 1, 121, 10, X (Voice, Notes, and Tone values may vary)
60 DATA (Data values may vary from 0 to 15 in any order.)

4. Rewrite the program so that ATARI plays Notes randomly. (HINT—drop the READ/DATA statements and use a LET statement to assign a random number to N.)

```
10 REM RANDOM SONG
20 LET N=INT(254*RND(1)+0)
30 SOUND 1, N, 10, 8
40 FOR T=1 TO 150:NEXT T
50 GOTO 20
```

PROGRAMMER'S PASTIME #65

Answers

1. Select a simple song from a music book (Mary Had a Little Lamb, Jingle Bells, etc.), and write a program so that the song can be played by ATARI. (HINT—Use this formula and set in the proper Notes, SOUND: 0, N, 10, 8. Later, vary the Voice, Tone, and Loudness to see what happens.)

Answers will vary. Teachers are encouraged to try their own.

2. Write a program so ATARI will play music that you have composed.

Answers will vary.

PROGRAMMER'S PASTIME #66

Answers

One of the most important aspects of producing good graphics, is being able to place the points and lines exactly where you want them. The key to doing so is to exactly locate a point by its column (X Coordinate) and row (Y Coordinate) position.

Use graph paper or the graphics screen illustrations that are on the next pages. Locate the following points on the Graphics 3 and Graphics 6 & 7 screens. Check your answers on ATARI. (Note — Although the two worksheet screens look similar, the numbering systems are different.)

Graphics 3	Graphics 7	
1, 10	1, 10	A
10, 1	10, 1	B
0, 0	0, 0	C
5, 19	5, 19	D
0, 19	0, 19	E
19, 16	0, 79	F
39, 0	159, 0	G
39, 19	159, 79	H
15, 22*	60, 40	I
10, 24**	76, 161**	J
	76, 197**	K

* This would be in the text windows and would not be visible.

** This cannot be plotted on the Graphics screen.

Because of the small size of the Graphics 7 screen, the locations will be shown as letters A-K on the graph.
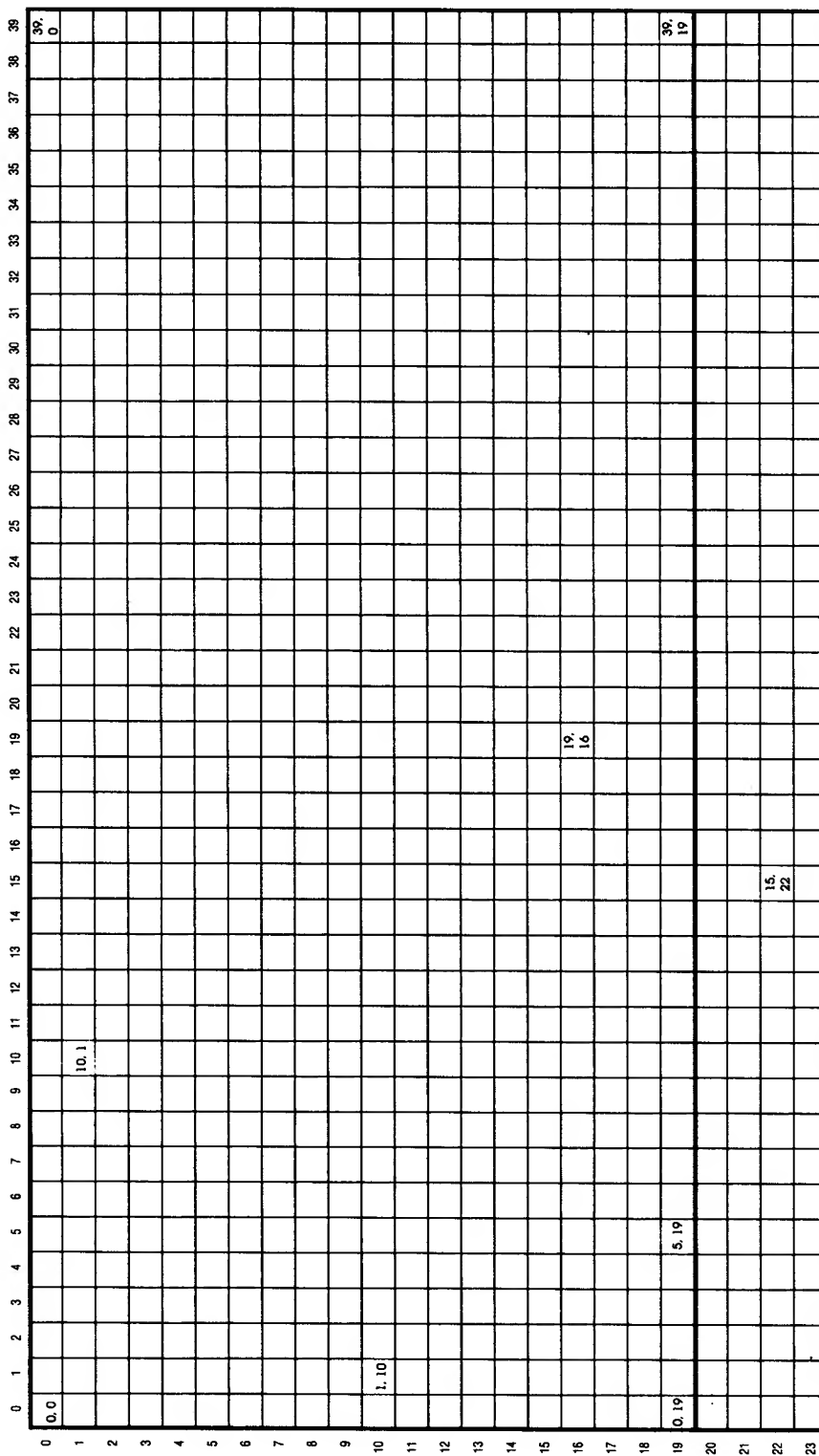
GRAPHICS 3 SCREEN
(20×40—WITH TEXT WINDOW)
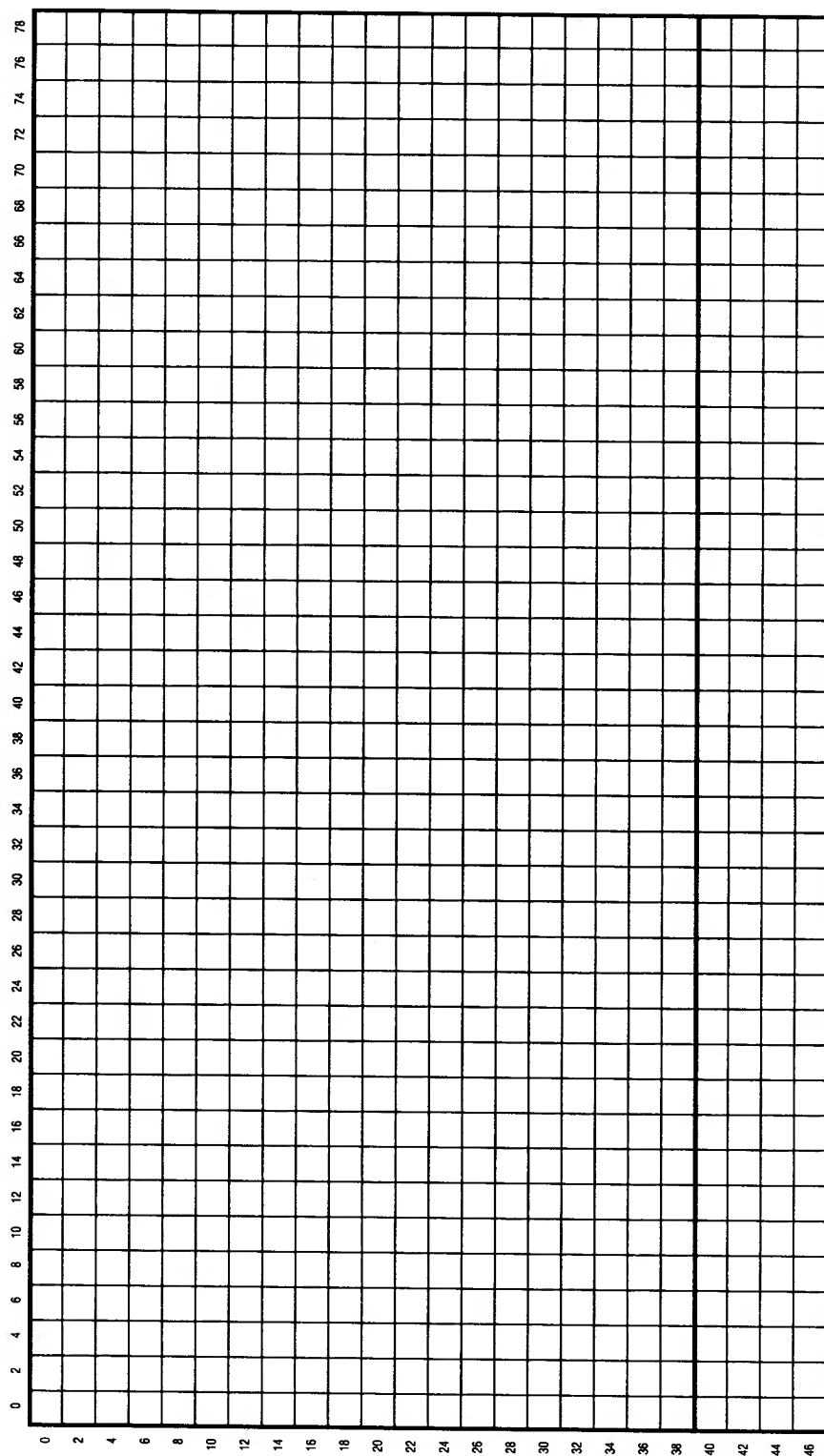
GRAPHICS 4 & 5 SCREEN
(40×80—WITH TEXT WINDOW)
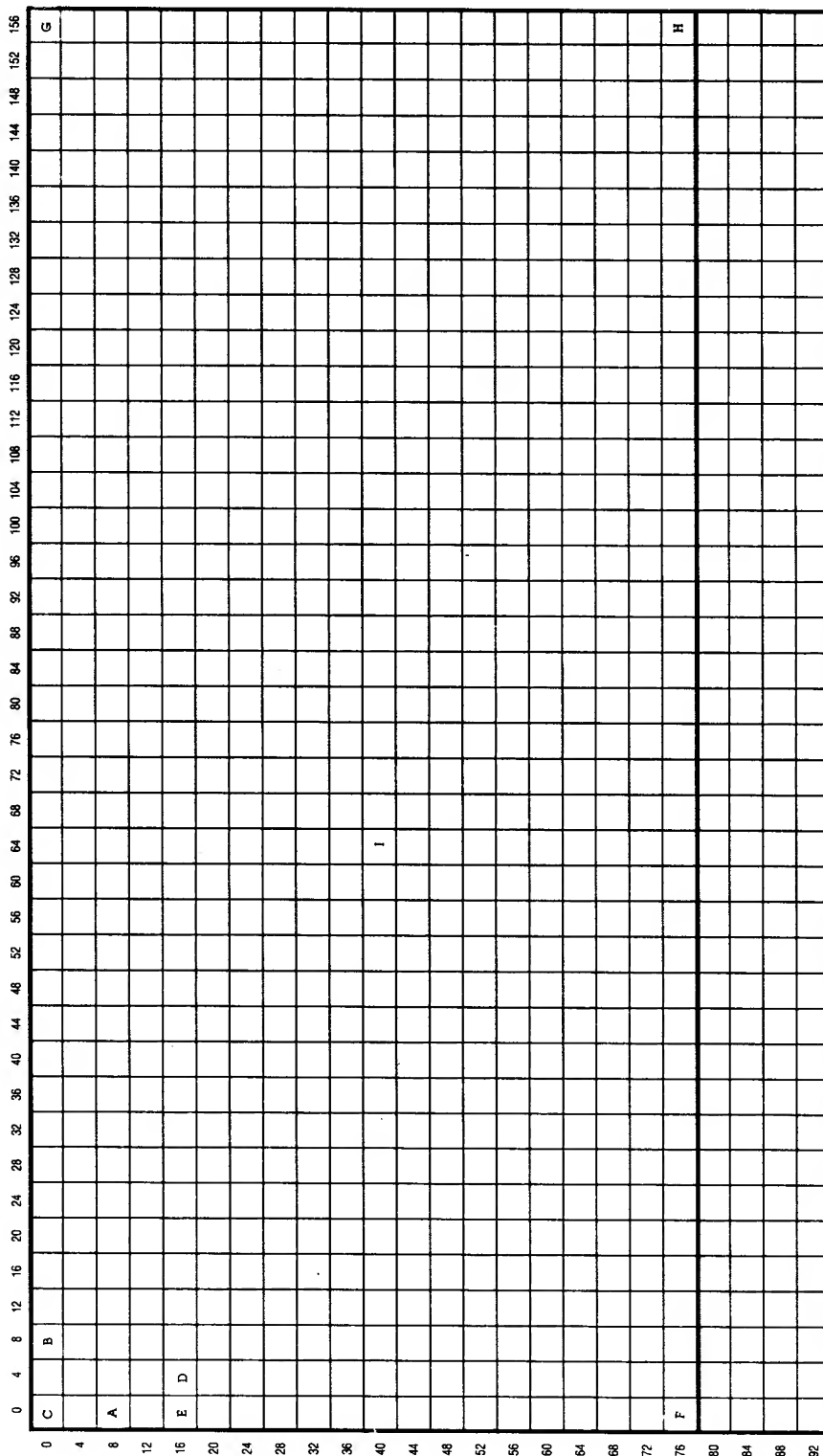
GRAPHICS 6 & 7 SCREEN
(80×160—WITH TEXT WINDOW)

Answers

One of the most enjoyable aspects of graphics is the ability to draw pictures. The key to doing so is to lay out a drawing on graph paper, and then convert the graph dimensions to statements that ATARI can understand. Notice the following drawing.

Columns (X)   Graphics 3 Screen

Rows (Y)

READY
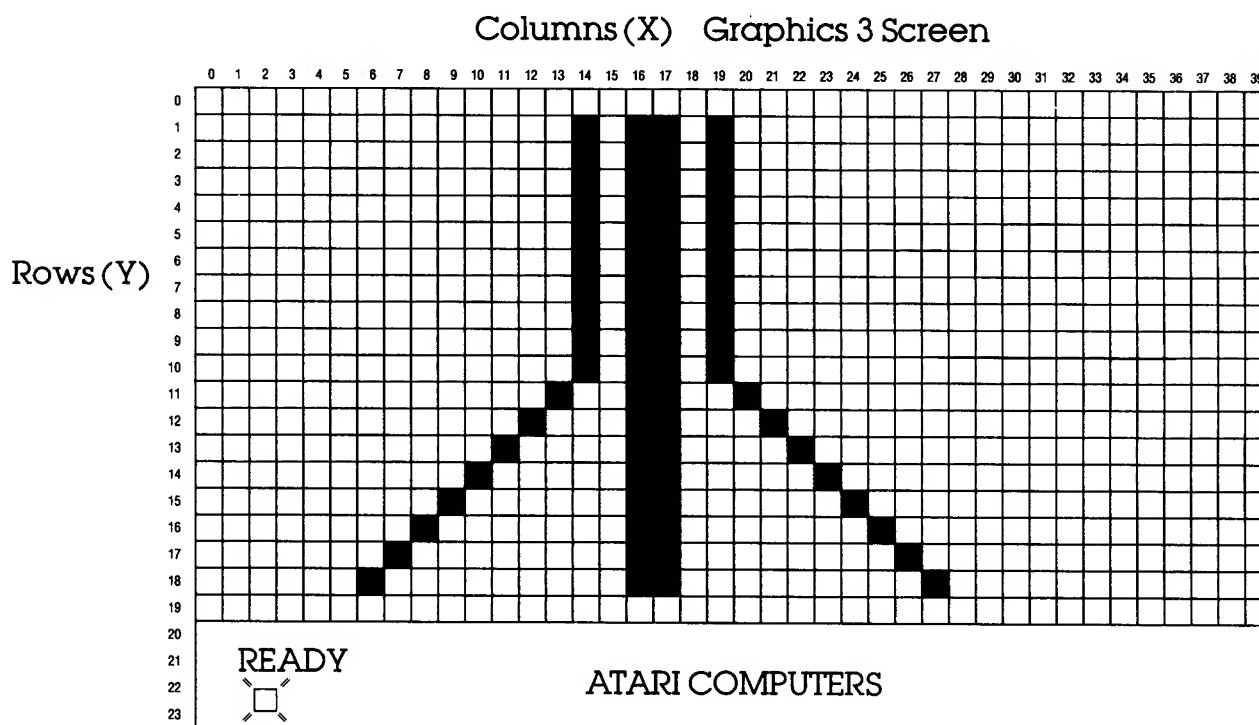
ATARI COMPUTERS

Here's how the drawing can be programmed for ATARI to understand:

```
10  GRAPHICS 3
20  COLOR 3
30  PLOT 14,1
40  DRAWTO 14,10
50  DRAWTO 6,18
60  PLOT 16,1
70  DRAWTO 16,18
80  PLOT 17,1
90  DRAWTO 17,18
100  PLOT 19,1
110  DRAWTO 19,10
120  DRAWTO 27,18
130  ?"        ATARI COMPUTERS"
```

206

1. Try modifying the above program by adding COLOR statements at various lines, and by changing some of the DRAWTO and PLOT statements.

2. Get several pieces of graph paper from your teacher, or a reuseable piece that has been laminated, and draw some pictures. Convert the drawings to programs, and try them on ATARI. (It's often easiest to begin with simple drawings for the Graphics 3 screen. Just be sure that you use the appropriate graph paper for the Graphics Mode you want to use.)

Answers will vary.

One enjoyable aspect of graphics is animation —causing the graphics to move. Following is a program for some simple animation.

```
10  GRAPHICS 3
20  COLOR 1
30  FOR X=0 TO 39 STEP 3
35  ?#6;" [ESC] [SHIFT] [CLEAR<] "
40  PLOT X,7
50  DRAWTO X,10
60  DRAWTO X+3,10
70  DRAWTO X+3,7
80  DRAWTO X,7
85  FOR T=1 TO 100:NEXT T
87  IF X> =36 THEN GOTO 10
90  NEXT X
```

Here's what the program does. Lines 40 through 80 make a simple square graphic. The X Coordinate is not specified in these lines, but rather it is set as a variable X. Lines 30 and 90 make the X Coordinate as every third number between 0 and 39. These lines, along with 10 and 20, which determine the graphics mode and color, are the main part of this program. However, notice how the program was improved by adding some more lines after the program was first written. Line 85 is a "timer" so that the graphic remains momentarily on the screen. (Try changing this line for different effects.) Line 87 causes the program to repeat once the graphic has moved completely across the screen. Line 35 causes the screen to be cleared as the graphic starts over. (Remember, #6 must be used with a PRINT statement for the graphics screen!)

1. Run this program, and then modify some line statements to see how you can change the graphics and animation.

2. Use all the graphic techniques that you have learned to this point to make your own animated graphics.

Answers will vary.

PROGRAMMER'S PASTIME #69

Answers

1. It takes a lot of practice to know all the graphic variations you can create with ATARI. Using the following program, experiment by changing the COLOR, GRAPHICS MODE, and SETCOLOR factors.

```
10 GRAPHICS 3
20 COLOR 1
30 FOR X=0 TO 15
40 SETCOLOR 0,X,2
50 PLOT 5,5
60 DRAWTO 25,5
70 FOR T=1 TO 600:NEXT T
80 ?:?:?X
90 NEXT X
100 END
```

2. Take some graphic programs you have already written, or make some new ones, and improve them by using the SETCOLOR statement.

Answers will vary.

PROGRAMMER'S PASTIME #70

Using all the techniques you have learned for graphics, sound, and regular programming, create a fantastic light and sound show!

Answers will vary.

Answers

Use what you know about a good game program to write the game programs described below.

1. It is more meaningful to the user when the computer calls him or her by name—it makes the interaction more personal.

Write a GUESS A NUMBER game program that asks the user's name and calls the user by name throughout the program.

1. THINK about the program
2. DATA TABLE
   Input variables
   N$=your name
   G=number guessed
   A$=answer to question, "Want to play again"
   Program variable
   X=random integer between 1 and 100

3. ALGORITHM
   1. Ask the player for name.
   2. Create a random integer.
   3. Ask player to guess.
   4. Compare guess with random integer.
   5. If guess is incorrect, tell why it is incorrect, and ask the player to guess again.
   6. If guess is correct, congratulate the player and ask if they want to play again.

4. Flow Chart

START

DIMENSION VARIABLES

? "GUESS A NUMBER GAME" : ? : ? "WHAT IS YOUR NAME";

INPUT N$

LET X = INT(100 * RND(1)+1)

? : ? "GUESS A NUMBER,b" N$ "BETWEEN 1 AND 100"

INPUT G

IS G = X ? — YES →

NO ↓

IS G > X ? — YES →

NO ↓

? "TOO LOW,b" N$

? "TOO HIGH,b" N$

? : ? "YOU GUESSED IT, b"; N$? "GOOD WORK!" ? : ? "WANT TO PLAY AGAIN";

INPUT A$

DOES A$ = "YES" ? — YES

NO ↓

? : ? "THANKSb" N$? "HAVE A GOOD DAY!"

STOP
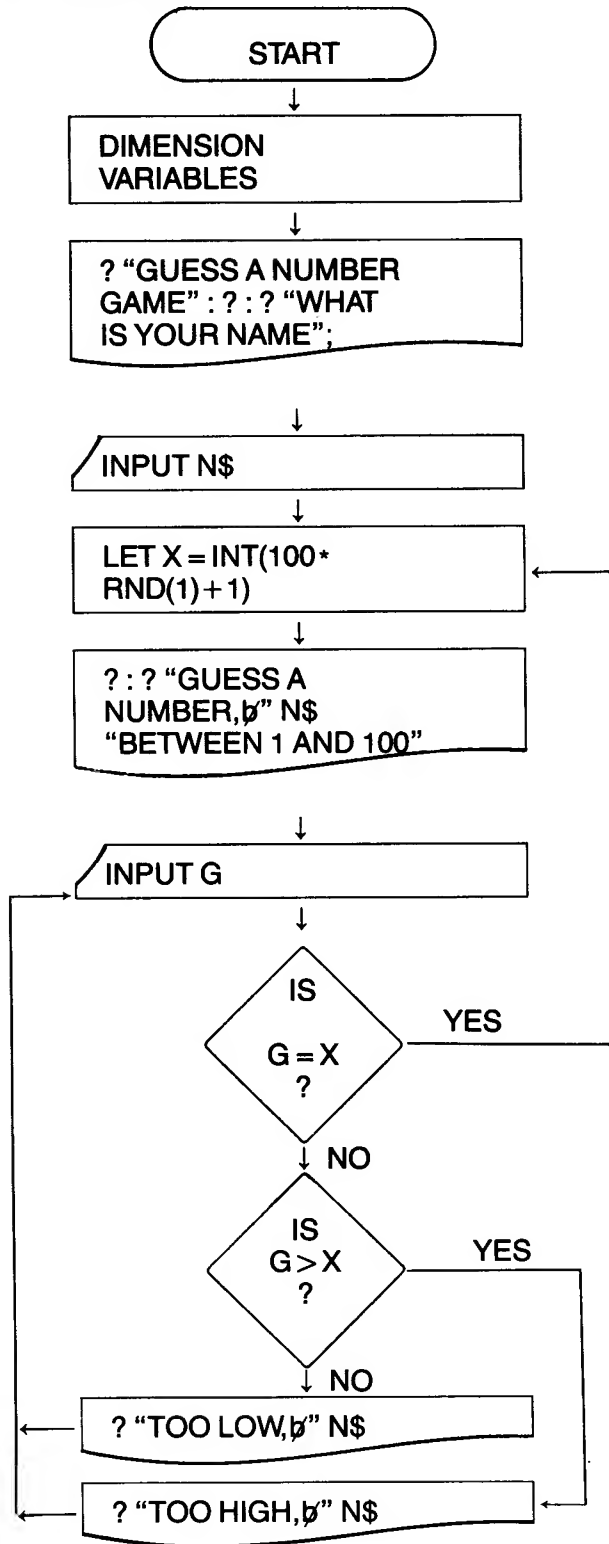
5. CODE

```
 10  DIM N$(30),A$(10)
 20  ? "GUESS A NUMBER GAME"
 30  ? : ? "WHAT IS YOUR NAME";
 40  INPUT N$
 50  LET X=INT(100*RND(1)+1)
 60  ? : ? "GUESS A NUMBER,b"; N$
 70  ? "BETWEEN 1 AND 100"
 80  INPUT G
 90  IF G=X THEN 120
100  IF G>X THEN ? "TOO HIGH,b"; N$:
     GOTO 80
110  IF G<X THEN ? "TOO LOW,b"; N$:
     GOTO 80
120  ? : ? "YOU GUESSED ITb"; N$
130  ? "GOOD WORK!"
140  ? : ? "WANT TO PLAY AGAIN";
150  INPUT A$
160  IF A$= "YES" THEN 50
170  ? : ? "THANKSb"; N$
180  ? "HAVE A GOOD DAY!"
190  END
```
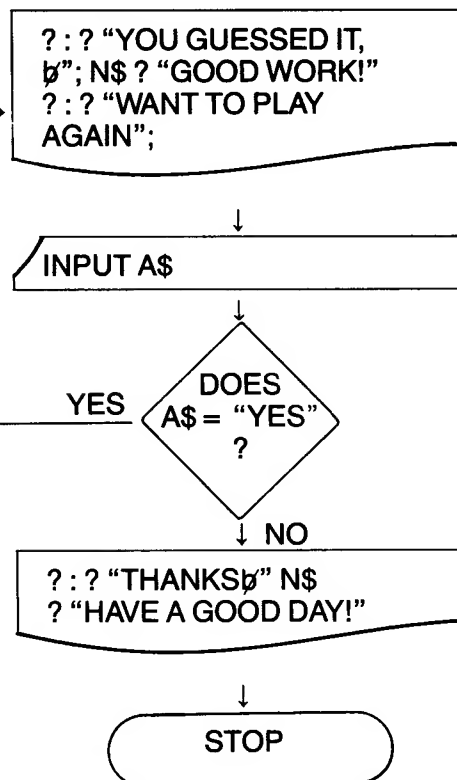
6. DEBUG
7. REVISE

2. Revise the game program in #1 so a player must answer "YES" or "NO" in line 150.

If anything else is typed for INPUT, make ATARI print the question in LINE 140 again. This helps make the program GOOF PROOF.

```
170 IF A$="NO" THEN 190
180 GOTO 140
190 ?:? "THANKS FOR PLAYING ʁ"; N$
200 ? "HAVE A GOOD DAY!"
210 END
```

3. Add something to the game program in #2 so ATARI tells the user how many tries it took before they guessed the correct number.

HINT: Use a COUNTER, C.

Set C at 0 before the first guess. After the first guess add 1 to the counter: C=C+1.

Then after each of the next guesses, make sure one more is added to the counter. When the correct number is guessed, make ATARI print IT TOOK YOU _____ GUESSES.

```
 75 LET C=0
 85 LET C=C+1
125 ? "IT TOOK YOUβ";C;"βTRIES"
```

4. Add something to the game program in #3 so
   ATARI asks the user the top number in the
   range they wish to guess. (For example, 1 to
   ____?)

After the user types in the top number, use it in the
RND function to create a random integer be-
tween 1 and the top number.

Hint: IF N=the top number
      THEN you would use this RND function:
          LET X= INT(N*RND(1)+1)

```
42  ?: ? "WHAT IS THE HIGHEST NUMBER"
44  ? "YOU WISH TO GUESS";
46  INPUT TN
50  LET X=INT(TN*RND(1)+1)
70  ? "BETWEEN 1 AND◊";TN
160 IF A$="YES" THEN 42
```

5. Make up a computer game that uses a die. Write a program using all of the good style techniques you've learned.

The RND function for the throw of your die must choose a random integer between 1 and 6.

Example
1. THINK about the program.
2. DATA TABLE
   Input variable
   Q$=answer to question, "WANT TO PLAY AGAIN"

   Program variables
   X=random integer representing throw of dice
   T=FOR–NEXT time loop

3. ALGORITHM
   1. Print game directions.
   2. Create a random integer between 1 and 6 (roll of die).
   3. Compare roll to 2 or 3.
   4. If it is a 2, tell the player he/she has lost.
   5. If it is a 3, tell the player he/she has won.
   6. Ask if they want to play again.
   7. If it is neither 2 nor 3, keep rolling.

4. Flow Chart
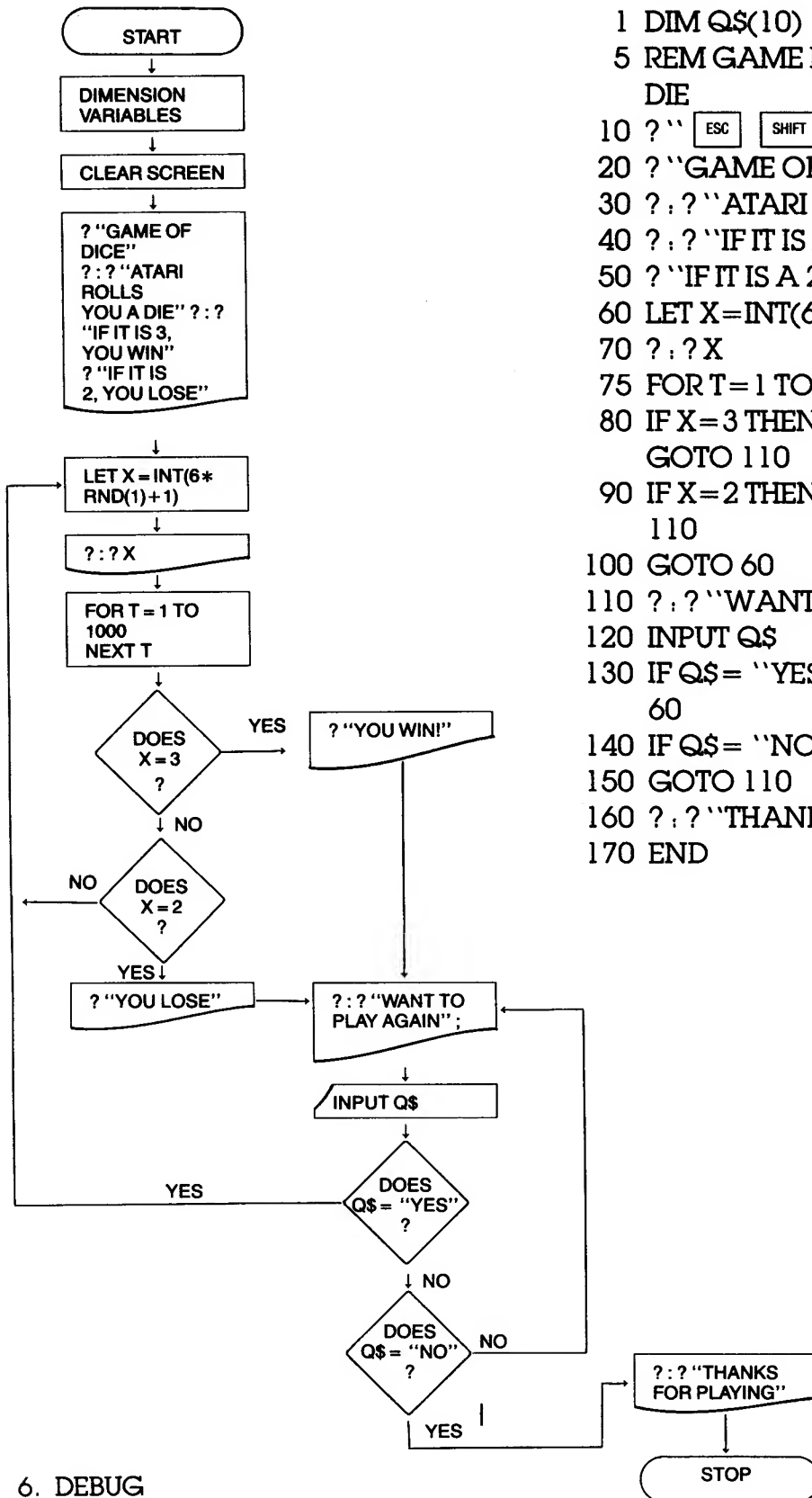
5. CODE

```
  1 DIM Q$(10)
  5 REM GAME PROGRAM USING A
    DIE
 10 ? " [ESC] [SHIFT] [CLEAR <] "
 20 ? "GAME OF DICE"
 30 ? : ? "ATARI ROLLS YOU A DIE"
 40 ? : ? "IF IT IS A 3, YOU WIN"
 50 ? "IF IT IS A 2, YOU LOSE"
 60 LET X=INT(6*RND(1)+1)
 70 ? : ? X
 75 FOR T=1 TO 1000 : NEXT T
 80 IF X=3 THEN ? "YOU WIN!" :
    GOTO 110
 90 IF X=2 THEN ? "YOU LOSE" : GOTO
    110
100 GOTO 60
110 ? : ? "WANT TO PLAY AGAIN";
120 INPUT Q$
130 IF Q$= "YES" THEN? "   " : GOTO
    60
140 IF Q$= "NO" THEN 160
150 GOTO 110
160 ? : ? "THANKS FOR PLAYING"
170 END
```

6. DEBUG
7. REVISE

217

6. Create a computer program for any game you like. Include the five things every good game program should have. Be creative!

Programs will vary.

1. THINK about the program
2. DATA TABLE

3. ALGORITHM

4. Flow Chart   5. CODE the program

6. DEBUG
7. REVISE

COMPONENT 7 FUN PAGE

Answers
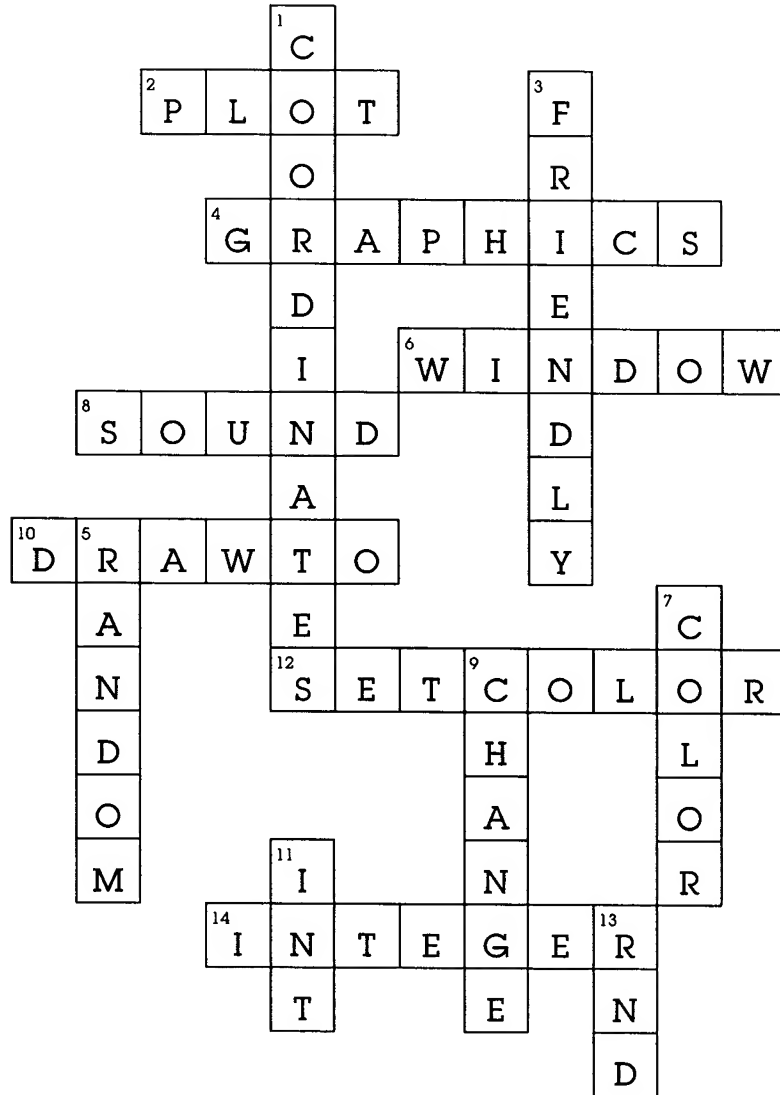
Crossword puzzle answers:

- 1. CODORDINATE (down: C-O-O-R-D-I-N-A-T-E)
- 2. PLOT (across)
- 3. FRIENDLY (down)
- 4. GRAPHICS (across)
- 6. WINDOW (across)
- 7. COLOR (down)
- 8. SOUND (across)
- 9. CHANGE (down)
- 10. DRAW TO (across)
- 5. RANDOM (down)
- 12. SET COLOR (across)
- 11. INT (down)
- 13. RND (down)
- 14. INTEGER (across)

PROGR PAST. 29
30
31
32
33
34
35
36
37
COMP. 5 FUN?
EVAL. YOURSELF
PROGR PAST. 38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
COMP. 6 FUN?
EVAL. YOURSELF

PROGR PAST 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | COMP ? FUN ? | EVAL. YOURSELF

KEYBOARD ILLUSTRATIONS

ATARI 800

ATARI XL

Index

This teacher's guide accompanies the book **AN ATARI FOR KIDS** and the student workbook, **AN ATARI in the Classroom: Activity Workbook**. This helpful guide:

- Outlines the behavorial objectives
- Includes complete lesson plans
- Answers all student worksheet questions and tests in the Activity Workbook
- Provides extra information useful to teachers, plus notes on how to use the curriculum
- Has additional information and hints on converting examples to other micro-computer models

AN ATARI FOR KIDS and **AN ATARI in the Classroom: Activity Workbook** also available.

MARTY URMAN 1983